DECOMPOSITION OF LARGE-SCALE SINGLE-COMMODITY NETWORK FLOW PROBLEMS

A THESIS

Presented to

The Faculty of the Division of Graduate Studies

By

Suleyman Tufekci

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy in Industrial and Systems Engineering

Georgia Institute of Technology

August, 1977

To the Memory of My Father

DECOMPOSITION OF LARGE-SCALE SINGLE-COMMODITY

NETWORK FLOW PROBLEMS

Approved:

John J. Jarvis, Chairman

Gary R. Parker

Leon F. McGinnis

Atif S. Dabs, E.E.

Matthew J. Sobel, I.M.

Date approved by Chairman: 9/2/77

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to Dr. John J. Jarvis, my dissertation advisor whose insight and comments contributed not only to the formulation of the algorithms, but also to my appreciation of the independent research effort.

Special thanks are also due to Dr. Gary R. Parker, Dr. Leon F. McGinnis, Dr. Atif S. Debs, and Dr. Matthew J. Sobel, who served on my reading and/or guidance committee and provided many helpful recommendations.

The author would also like to express his thanks to Dr. Robert N. Lehrer, Director of the School of Industrial and Systems Engineering and to Dr. William W. Hines, Director of Graduate Studies for their understanding, assistance, and help which allowed me to pursue full-time studies.

Finally, and most importantly, I especially wish to thank Ms. Donna I. Leibowitz, who typed and edited several drafts of this manuscript. Without her understanding and moral support the successful completion of this research would not have been possible.

The author expresses his sincere thanks to Ms. Shirleye McCombs who sacrificed her weekend to type the final version of this dissertation.

NOMENCLATURE

$G = (N,A)$	network with node set N and arc set A
N	node set of a network
A	arc set of a network
i	node index
(i,j)	an arc from node i to node j
$\|N\|$	number of elements in the set N (the number of nodes)
$\|A\|$	the number of arcs in the arc set A
m	number of subnetworks
u	number of nodes in each subnetwork
v	number of nodes shared by two adjacent subnetworks
X	a subset of the node-set N
$\bar{X}$	Complement of X, $\bar{X} = N - X$
f	amount of flow in the network
u_{ij}	upper capacity of flow in arc (i,j)
f_{ij}	flow on arc (i,j) corresponding to flow f in G
s	source node
t	sink node
e_k	maximum allowable flow that can be transferred through node k.
f^*	the maximum flow
d_{ij}	the distance (weight) between nodes i and j
D^0	original distance matrix (n x n)
D^*	matrix of shortest distances (n x n)

NOMENCLATURE (continued)

d_{ij}^{k}	the shortest distance between nodes i and j in the k^{th} iteration
x_{ij}	variable corresponding to flow in arc (i,j)
w_i'	label on node i
N_j	subset of $\bar{N}_j$ such that if $i \epsilon \bar{N}_j$ and $i \notin \bar{N}_k$ for $j \neq k$
X_j	the set of nodes shared by two adjacent subnetworks $\bar{N}_j$ and $\bar{N}_{j+1}$
P	a path
$(X,\bar{X})$	set of arcs (i,j) such that $i \in X$ and $j \in X$ and $(i,j) \in A$
$c(X,\bar{X})$	capacity of the cut $(X,\bar{X})$
u_{ij}'	upper capacity of arc (i,j) in the marginal network
G'	marginal network
Δ	incremental flow
$(Z_i,\bar{Z}_i)$	a minimum cut for the subnetwork G_i
A_i	arc set of subnetwork G_i
$(Y,\bar{Y})$	min-source cut
s', s''	artificial sink (source)
s_k	artificial sink between subnetworks G_k and G_{k+1}
$(Y_k,\bar{Y}_k)$	the minimum cut for the network $\bigcup_{i=1}^{k} G_i$
S_k	increasing or decreasing sequence of min-cut values of p adjacent subnetworks
$d_{ij}^{*}(\bar{N}_1 \cup .. \cup \bar{N}_k)$	the conditional shortest distance between nodes i and j
d_{ij}^{*}	actual shortest distance between nodes i and j
$D_{XX}(N)$	actual shortest distance between all nodes of the distance matrix X

NOMENCLATURE (continued)

$\otimes$	matrix minisummation
X'	a new matrix formed from X by adjoining nodes s and t in it
$G'' = (N', A'')$	a new network obtained from the original network G by only taking the common nodes
N'	node set of network G''
$G' = (N', A')$	a network obtained from the original network G as in G''
A'	arc set of G', different from A''
G^f	marginal network obtained from the original network with respect to flow f
G^{fk}	marginal network obtained from the original network with respect to flow f^k at iteration k
π_i	labeling function on node i
C	a cycle
$\bar{d}_{ij}$	modified nonnegative weight on arc (i,j)
w_i^k	shortest distance from node s to node i at iteration k
$\bar{N}_k^f$	marginal subnetwork obtained from $\bar{N}_k$ with respect to flow f
$\bar{N}_j^{f^k}$	marginal subnetwork obtained from $\bar{N}_j$ with respect to flow f^k at iteration k
G''^{f^k}	marginal cost network obtained from network G'' w.r.t. flow f^k at iteration k

SUMMARY

Several network flow problems are solved by decomposition in this dissertation. Research is conducted in three important areas of network flow problems.

In the first area location of minimum cut(s) by decomposition is studied. With the aid of three lemmas developed, an algorithm for locating the min-cut(s) is proposed. It is also shown that the theoretical efficiency of the proposed algorithm (Algorithm A) is quadratically (in number of subnetworks) more efficient than a no-decomposition algorithm in the best case. In the worst case, the algorithm is shown to be more efficient than a no-decomposition algorithm in a probabilistic sense. The probability of superiority approaches to a unity for the number of nodes of a network exceeding 20.

Determining the shortest path from a source node to a sink node of a large-scale sparse network is studied as the second area. The proposed decomposition algorithm (Algorithm C) is basically composed of decomposing the original network into m linearly overlapping subnetworks and performing a shortest path algorithm on each subnetwork. Based on the information obtained from subnetworks a new acyclic network much smaller in size than the original network is formed. Finally a special shortest path algorithm (Algorithm B) is applied on this special network. It is shown that the proposed decomposition algorithm is theoretically superior to a no-decomposition algorithm as well as to similar decomposition algorithms.

In the third area the minimal cost network flow problems are studied. Two efficient flow augmentation algorithms are given for minimum cost maximum flow problems in a large scale single commodity network (Algorithm D and Algorithm E). It is also shown that the theoretical efficiencies of Algorithm D and Algorithm E are better than the theoretical efficiencies of the existing minimum cost flow algorithms.

CHAPTER I

INTRODUCTION

Background

The development of optimization techniques in network flow problems follows the same pattern as in linear optimization problems. Theories and algorithms were developed to solve small and moderate problems. After the discovery of those algorithms, they were applied to "real world" problems, more complex in their nature compared to the ones that led to the development of the theory.

Existing algorithms were incapable of handling the large-scale problems existing in the "real world". Even with the development of high-speed, large-memory digital computers, it was impossible to handle these problems with the available algorithms.

The idea of decomposing the large-scale problems to smaller subproblems and performing the optimization techniques on those subproblems was developed. A large number of decomposition algorithms have been developed for large-scale linear programming problems.

Since a network flow problem can be represented as a linear programming problem, all of those algorithms which were developed for linear programming problems were applicable to the network flow problems. In some cases those algorithms were much easier to apply to network flow problems due to the special structure of those problems.

Most of the decomposition algorithms developed for network flow

problems utilize the linear programming formulation as a basis. There have been a number of papers in the literature on decomposition of networks from a graph theoretic point of view.

Problem Statement

The research problem examined in this dissertation is the development of decomposition algorithms for various classes of single commodity network flow problems.

In the case of the location of min-cut(s) by decomposition, a network is decomposed linearly into m disjoint node-sets and Edmonds and Karp's [11] "first-labeled first-scanned" labeling algorithm is performed on each subnetwork to locate the actual min-cut for the original network. Additional labeling may be required for the union of some subnetworks depending upon the values of the minimum cuts obtained for each subnetwork. Naturally, the location of the min-cut(s) will give decomposition of the original network into two or more subnetworks, depending upon the number of min-cuts existing in the original network.

Once the network is naturally decomposed, further analyses (max-flow or min-cost max-flow) can be carried out on the subnetworks individually. The union of the optimal solutions obtained on the subnetworks will correspond to the optimal solution of the overall network.

In the case of the shortest path problem, the network is decomposed into m overlapping subnetworks as described by T. C. Hu [26]. The algorithm determines only the shortest path of the original network from the source to the sink. The algorithm developed for the shortest path will then be applied to the min-cost max-flow problem. At every flow augmen-

tation, the decomposition algorithm for the shortest path is used to determine the flow augmenting path.

Problem Importance

The operation of large firms and the responsibilities of local governments are increasing rapidly. National firms are becoming multi-national firms and governments are undertaking greater responsibilities in the areas of economics, welfare, food and energy. The Military has already become a giant organization. All of these organizations have one thing in common: the distribution of goods and/or services.

In multi-national companies, one of the major problems is the distribution of raw material and finished goods among a large number of demand points. Represented as a network flow problem (single-commodity or multi-commodity), the size of the network becomes almost impossible to handle with ordinary network flow algorithms.

From a governmental point of view, the distribution of food and energy are becoming two of the most critical issues that must be dealt with. Half of the world is starving. We need to find the best methods to produce and distribute food to the people. Another important national issue is energy. Energy resources are not unlimited. They must be used wisely (including both energy production and energy distribution).

In military operations, one of the most important and unsolved problems is the logistics problem. For example, the military is concerned with the problem of supplying at least 60-70 air bases from every other airbase, with spare parts and supplies every day. The military supply program can be viewed as either a single-commodity or multi-commo-

dity flow problem. The problem can be made more complicated by adding extra constraints, such as cargo plane constraints, container constraints, and delivery time constraints.

It can be concluded that many "real world" single-commodity network problems are very large in their nature. They need to be solved by a special method that will enable us to decompose these large scale network flow problems into smaller, more manageable subproblems.

Results

Chapter II gives an extensive literature review on network flow problems and shortest path algorithms. The shortest path algorithms are grouped into two categories; as matrix algorithms and as treebuilding algorithms. Section C of Chapter II gives the review of existing decomposition algorithms.

It is shown in Chapter III that the decomposition algorithm for the location of the min-cut(s) is almost always superior to a no-decomposition procedure. Theoretical upper bounds on the number of flow augmentations required by the decomposition as well as an example problem are given. It is shown that the theoretical upper bounds for the decomposition algorithm for locating the min-cut(s) is quadratically (in number of subnetworks) superior to a no-decomposition algorithm in the best case. In the worst case, it is also shown that the theoretical efficiency of the algorithm is superior to a no-decomposition procedure in a probabilistic sense.

Two shortest path algorithms are presented in Chapter IV. It is shown that the theoretical upper bounds for the proposed shortest path

algorithms are superior to the no-decomposition algorithms as well as the decomposition algorithms which determine all the shortest distances in a given network. It is also shown in Chapter IV that the theoretical efficiencies of the proposed two algorithms are almost equal to each other.

Since the efficiency of the flow augmenting algorithms is a linear function of the efficiency of the shortest path algorithms they utilize, and since the proposed shortest path algorithms are theoretically superior to the no-decomposition or decomposition counter-parts, the min-cost max-flow algorithms proposed in Chapter V are proven to be theoretically more efficient than existing flow augmenting algorithms. On one of the algorithms proposed the flow augmentation is performed on a nonnegative cost, marginal cost network which is much smaller in size than the original network. It is also shown that this algorithm is as efficient, but not better, than the previous algorithm proposed in the same chapter.

In Chapter VI, theoretical upper bounds of the min-cost max-flow algorithms are reexamined. A new algorithm for determining the conditional shortest distances on each subnetwork is also given in Chapter VII. With the help of this algorithm, the theoretical upper bound on each flow augmentation is reduced to $O(5mu^2v)$ from $O(mu^3)$, where $u \gg v$. Hence, for $u > 5v$, the new algorithm is expected to determine the flow augmenting path faster than the previously proposed min-cost max-flow algorithms in Chapter V.

Chapter VII concludes with the results of this dissertation. Several recommendations are given for further research in the same area.

CHAPTER II

LITERATURE REVIEW

There has been considerable development in network theory since the string model [36]. Ford and Fulkerson [16] developed the max-flow min-cut theorem as well as the state-of-the-art out-of-kilter algorithm, shortest path algorithm and other major theorems.

In the late fifties and sixties a large number of publications appeared in the literature on shortest path algorithms. Minimal cost flow algorithms followed the same pattern. Flow augmenting algorithms (dual algorithms) and the algorithms that utilize the available maximum flow to detect negative cycles in the marginal cost network (primal algorithms) now compete with the state-of-the-art out-of-kilter algorithm.

In the mid sixties T. C. Hu [26] was the first to state a decomposition algorithm for finding the shortest distances between all pairs of nodes. It is followed by Yen's [51] modification. Glover, Klingman and Napier [18] were the last to iterate this decomposition a step further to reduce the number of required computations.

In the late sixties and early seventies several decomposition algorithms were proposed on special structured networks [5], [30], [43], and [48].

A. Maximum Flow Minimum Cut

Ford and Fulkerson [16] were the first to prove the max-flow min-cut theorem which states that; for a given flow f, a cut $(X,\bar{X})$ is

minimal if, and only if all the arcs of $(X,\bar{X})$ are saturated while the arcs in the form $(\bar{X},X)$ are flowless with respect to f.

The following labeling algorithm is given by Ford and Fulkerson [16] to find the maximum flow (and minimum-cut) in a given general network G = (N,A); Every node in G is always in one of the three states, labeled scanned, labeled unscanned or unlabeled. Initially all nodes are unlabeled. First, node s gets labeled as $[s^+, e_s = \infty]$. The first entry of the label tells from which node this current node is labeled and the second entry tells how much flow can be sent from the labeling node to the labeled node.

In general, the algorithm proceeds by selecting labeled and unscanned node j and assigning the label $[j^+, e_k]$ to all neighboring nodes k of j which are unlabeled and have $0 \leq f_{jk} \leq u_{jk}$. Where, $e_k = \min[e_j, u_{jk} - f_{jk}]$. Similarly, to all neighboring nodes k of j which are unlabeled and have $f_{kj} > 0$ assign the label $[j^-, e_k]$, where $e_k = \min [e_j, f_{kj}]$. If all the neighboring nodes of j have labels, then j is considered to be labeled and scanned. This process continues until (i) node t is labeled, or, (ii) node t cannot be labeled. In the case (i), the nodes are traced on the flow augmenting path, starting from t and ending in s. The flow is increased along the path by an amount e_t. All labels are erased and labeling starts all over again. In the case (ii) no flow augmenting path exists. Hence, current flow is optimal. The set X of the labeled nodes and the set of unlabeled nodes $\bar{X} = N - X$ denotes the minimum cut for the network.

Edmonds and Karp [11] developed the following refinement on the labeling procedure. At any step of the flow augmentation relative to a

given flow f in G = (N,A) a flow augmenting path is defined as a path such that

(a) if $(i,j) \in A$ and $(j,i) \notin A$ then $e_i = u_{ij} - f_{ij} > 0$

(b) if $(i,j) \notin A$ and $(j,i) \in A$ then $e_i = f_{ji} > 0$

(c) if $(i,j) \in A$ and $(j,i) \in A$ then $e_i = u_{ij} - f_{ij} + f_{ji} > 0$

For a given flow augmenting path P let $e = \min e_i > 0$. Alter the flow along P as follows: (a) increase flow on arc (i,j) by e; (b) decrease the flow on arc (i,j) by e; (c) increase the flow on arc (i,j) by $\min (e, u_{ij} - f_{ij})$ and decrease the flow on arc (j,i) by $\max (0, e - u_{ij} + f_{ij})$. The algorithm differs from Ford and Fulkerson's in step (c) which allows sending the flow in both direction (i,j) and (j,i), simultaneously. The algorithm scans on a "first-labeled first-scanned" basis. Under this condition the maximum flow will be achieved after no more than $\frac{1}{4}(n^3 - n)$ flow augmentations. If the augmentation is done along the path with the maximum flow capacity, then the number of flow augmentations is bounded by $1 + \log_{m(m-1)} f^*$, where $m = \max_{X \subseteq N - \{t\}} |(X,\bar{X})|$ and f is the value of the maximum flow.

The linear programming formulation of the maximum flow problem is given by Dantzig [8] and Ford and Fulkerson [16]. If f is the variable designating the flow in the network, then the linear programming formulation of the maximum flow problem is:

maximize f

$$\text{S.T.} \quad \sum_{i=1}^{n} x_{ij} - \sum_{k=1}^{n} x_{ki} = \begin{cases} f & i = 1 \\ 0 & i \neq 1,\ i \neq m \\ -f & i = m \end{cases}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad i, j = 1, \ldots, n$$

The totally unimodular property of the constraint matrix guarantees an integer optimal solution.

Gomory and Hu [20] gave an algorithm for determining maximal flow values between several nodes of a given undirected (symmetric) network. The process regards a subset of nodes of the network as a single node (which is equivalent to assuming the added arcs between every pair of nodes of the subset with infinite capacity). The arcs directly connecting a node i, not in the subset, to any nodes in the subset are replaced by a single arc having a capacity equal to the sum of the capacities of the connecting arcs. The maximum flow computations are then carried on this condensed and smaller sized network. The process constructs a flow equivalent network, which is a tree. Each link (i,j) of the tree represents a minimum cut of the original network between nodes i and j. This tree is called a cut-tree. A cut-tree of n nodes shows the n-1 minimum cuts of the original network which do not cross each other.

B. Shortest Path

Many algorithms have been proposed for finding the shortest path in a network since Minty's [36] string model. From the point of view of practicality, only a handful of them are proven to be computationally efficient [10]. The algorithms can be grouped into two major categories, as given by Steenbrink [45], as the matrix algorithms and the tree-building algorithms.

Matrix Algorithms

The algorithms which fall in this class obtain shortest paths between all the nodes simultaneously. The actual length of the shortest paths and the paths themselves are stored in two (n x n) matrices, where n is the number of nodes in the network.

The algorithm of Floyd [15] starts with the original distance matrix D^0 and changes it into the final matrix D^* of shortest distances in n steps. Generally, in the k^{th} step the following "triple" operation is used.

$$d_{ij}^{k} = \min \left\{ d_{ij}^{k-1} \ , \ d_{ik}^{k-1} + d_{kj}^{k-1} \right\}$$

for all $i \neq j \neq k$. For each node i of the network, the preceeding nodes are kept in an (n x n) matrix, in which every row or column forms a tree. The algorithm requires n(n-1)(n-2) additions and comparisons to form the final distance matrix D^*.

Another matrix algorithm is proposed by Dantzig [7] in which n matrices are generated successively, where the n^{th} matrix contains the shortest distances between all pairs of nodes. It starts with the matrix $D^0 = d_{11}$ of size (1 x 1) and obtains the matrices of one higher order until the complete distance matrix D^* of size (n x n) is obtained, where d_{11} is the first entry of the original distance matrix. The operations in the k^{th} step are as follows:

$$d_{ik}^{k} = \min_{i \leq j \leq k-1} \left\{ d_{ij}^{k-1} + d_{jk} \right\} \text{ for all } 1 \leq i \leq k-1$$

$$d_{ki}^{k} = \min_{1 \le j \le k-1}\left\{d_{kj} + d_{ji}^{k-1}\right\} \text{ for all } 1 \le i \le k-1$$

and $d_{ij}^{k} = \min\left\{d_{ij}^{k-1}, d_{ik}^{k} + d_{kj}^{k}\right\}$ for all $1 \le i \le k-1$ and for all $1 \le j \ne i \le k-1$.

The number of additions and comparisons required by the algorithm is given by

$$\sum_{k=1}^{n} \{2(k-1)(k-1) + (k-1)(k-2)\} = n(n-1)^2$$

The algorithm of Yen [52] finds all the shortest distances from a fixed origin by considering the fact that any path from s to t contains the combination of m homogeneous blocks $1 < m \le n-1$, in which the numbers naming the nodes in each block either form a strictly increasing or decreasing sequence. The functional equations of the algorithms are as follows:

Set $d_i^o = d_{in}$ $\quad i = 1, \ldots, n$. Compute

$$d_i^{(2k+1)} = \min_{n \ge j > i}\left\{d_j^{(2k-1)} + d_{ij}, d_i^{(2k-2)}\right\}$$

$i = n-1, n-2, \ldots, 1$ $\quad d_n^{(2k-1)} = d_n^{(2k)}$. Compute

$$d_i^{(2k)} = \min_{1 \le j < i}\left\{d_i^{(2k)} + d_{ij}, d_i^{(2k-1)}\right\} \quad i = 2,3, \ldots, n$$

$$d_1^{(2k)} = d_1^{(2k-1)} \quad \text{for } k = 1,2, \ldots$$

The algorithm terminates when $d_i^{2k} = d_i^{2k-1}$ or $d_i^{2k+1} = d_i^{2k}$ $i = 1, \ldots, n$ The number of additions and comparisons required by the algorithm is given by $(1/2)m(n-1)(n-2)$, $1 < m \leq n-1$.

By exploiting the fact that at each iteration one additional d_i becomes permanent and hence does not affect the further calculations thereafter, Yen [53] modifies his algorithm to save (n-2) additions and comparisons in each of every two iterations. This modification led to the upper bound $0(1/4n^3)$ on the number of additions and comparisons, when $m = n-1$. Since this modification needs an additional $1/4n^3$ sortings, the new algorithm does not differ from the original algorithm much in terms of number of additions and comparisons involved.

Dantzig [8] proposes a simplex algorithm for the shortest path problem. The shortest path problem can be formulated as a linear programming problem as;

$$\text{minimize} \quad \sum_{i,j} c_{ij} x_{ij}$$

$$\text{S.T.} \quad \sum_{j=1}^{n} x_{ij} - \sum_{k=1}^{n} x_{ki} = \begin{cases} 1 & \text{if } i = 1 \\ 0 & \text{if } i \neq 1, i \neq m \\ -1 & \text{if } i = m \end{cases}$$

$$x_{ij} \geq 0 \qquad i,j = 1, 2, \ldots, n$$

Bennington [3] refined the simplex algorithm by showing that every arboroscence centered around the source node is a basic solution but the converse is not true. By restricting the basic feasible solutions to the set of arboroscences it is shown that the algorithm can reach the optimal solution in fewer steps, since the number of arboroscences are

less than the number of basic feasible solutions. Arboroscence is defined as the tree which spans all the nodes of the network.

Treebuilding Algorithms

In this group of algorithms the shortest paths from a fixed origin to all other nodes are determined.

The algorithm, due to Moore [37] originally sets all the distances d_{sj} to infinity (where s is the origin node). The algorithm proceeds to build the tree by assuming the origin to be "active" and excuting the basic operation $d_{sj} = \min_{(i,j)\epsilon A}(d_{si} + d_{ij})$. At each iteration, every node connected with an active node by an arc is checked to see if the existing distance can be replaced by a shorter one. If so, the node for which the distance has been shortened becomes active and the process is continued until all the nodes become active once. The upper bound on the number of additions and comparisons is given by $(n-2)(n-1)^2$. The algorithm is originally proposed for networks with nonnegative distances but it may as well be applied to an arbitrary distance network (provided no negative cycles exist).

Credit belongs to Dijkstra [9] for his extremely efficient shortest path algorithm where all arc weights are required to be nonnegative. The algorithm assigns tentative labels, which are upper bounds on the shortest distances from the origin node to all other nodes. At each iteration, one node label becomes permanent. After the following main step

$$w_p' + d_{pq} = \min_{(i,j)\epsilon(X,\bar{X})}(w_i' + d_{ij}), \quad \text{set } w_q' = w_p' + d_{pq}$$

executed exactly n times, all node labels become permanent, where X is

the set of the permanently labeled nodes and $\bar{X}$ is the set of unlabeled nodes. Every time a node gets permanently labeled, it is placed in X and dropped from $\bar{X}$. The algorithm requires n(n-2)/2 additions and n(n - 1) comparisons.

Bazaraa and Langely [2] give the method of converting the arc weights in a network into the equivalent nonnegative ones by utilizing the dual of the linear programming formulation of the shortest path problem. The upper bound on the computations required may reach $O(n^3)$.

C. Decomposition Algorithms

T. C. Hu [26] in 1968 proposed the first decomposition algorithm for determining the shortest paths between all pairs of nodes in a sparse network. The algorithm begins by decomposing the network linearly into m overlapping subnetworks. The "triple operation"

$$d_{ik} = \min \left[d_{ik}, d_{ij} + d_{jk} \right] \text{ for all } i \neq j \neq k$$

is applied on (m-1) subnetworks, beginning with the first, successively, where the conditional shortest distance obtained in one network will replace the original distances of the succeeding network. This procedure is applied once more starting from the m^{th} subnetwork working towards the first subnetwork in a reverse order. This forward and backward stream of triple operations determines the actual shortest distances between all pairs of nodes which lie in the same subnetwork. In order to find the shortest distances between two nodes which do not lie in the same network, the triple operation is carried in a similar manner in the matrix form, referred to as the "matrix minisummation", until all the

shortest distances are determined.

Suppose each subnetwork shares v nodes with its neighboring subnetwork and that the number of nodes that are not shared by the others is u, and that the network is decomposed into m overlapping subnetworks. The algorithm requires $(2m-1)u^3 + (m^2+11m-15)u^2 + (m^2+11m-23)v^3$ additions and comparisons. It requires $(mu + (m-1)v)^3$ additions and comparisons if the same network is solved by Floyd's matrix algorithm. For m large, the ratio of the number of operation in decomposition to no decomposition is $\frac{1}{m}\left(\frac{u}{u+v}\right)$.

Yen [51] shows that in the application of the triple operations in the reverse order, in T. C. Hu's algorithm, we do not have to consider the overall subnetwork. He also shows that fixing one node in the overlapping section of the two adjacent subnetworks saves $(m-1)u^2 + (5m-8)u^2v + (8m-15)uv^2 + (5m-9)v^3$ additions and comparisons. The total number of additions and comparisons required by this modified algorithm amounts to $mu^3 + (m^2 + 6m - 7)u^2v + (2m^2 + 10m -20)uv^2 + (m^2 + 6m -14)v^3$, which is approximately half the number required in T. C. Hu's original algorithm.

Glover, Klingman and Napier [18] suggest a different decomposition scheme, based on their experience, that in practice, most of the actual network flow problems do not contain directed paths from any node to any other node. They further claim that such paths only exist for the node pairs in the same "region" of the network. The decomposition starts by choosing the first node set arbitrarily, and selecting an arbitrary node to be its origin. The procedure then fans out from node to node to find all nodes to which this origin node is connected by a directed path. If

this process does not exhaust all the nodes in the network, since the network is connected, some of the unreached nodes must have directed paths from them to some of the reached nodes. The procedure continues in this fashion until the desired decomposition is obtained.

The decomposition obtained by this procedure has the following properties: for odd numbered subnetworks there exist no directed paths between the common nodes with the subnetwork on the left and the common nodes with the subnetwork on the right. For even numbered subnetworks, such a path may exist and the left common nodes and the right common nodes may intersect.

The computational phase of this algorithm utilizes the forward stream of the triple operations used in T. C. Hu's [26] decomposition, for odd numbered subnetworks. After the original distances between the overlapping nodes have been replaced by the conditional shortest distances obtained in the previous step, the same triple operation is applied on the even numbered subnetworks. In the last step the shortest distances between an even numbered subnetwork and its right and left adjacent subnetworks are determined. The algorithm requires $mu^3 + 7mu^2 + 2mu^2 + 13muv^2 + 7muv + 8mv^3 + 4mv^2$ additions and comparisons.

Although the number of operations needed by this algorithm looks impressive in the lower order terms, the need for the special structure shades the success of the algorithm. Obviously, if the sink in a given network can be reached from every other node, this algorithm produces no decomposition.

Shier [43] gives a decomposition algorithm for optimality problems in tree-structured networks. The node-set N of the given network is

partitioned into $m \geq 2$ disjoint sets of nodes $N_1, N_2, \ldots N_m$ which exhibit a tree-structure when viewed as an undirected graph. After the node-incidence matrix is partitioned accordingly, a "binoid" operation is performed on the submatrices. Different entities (shortest path, longest path, etc.) are determined depending upon the binoid operation used.

Werra [48] gives the decomposition of bipartite graphs into matchings. He shows that a bipartite multigraph has either a uniform or a nearly uniform decomposition into any number $m^* \geq m$ of matchings, where m is the maximal degree of the vertices in a bipartite network.

Brucker [5] gives a decomposition algorithm for shortest paths in a network with many strongly connected components. A network is partitioned in such a way that two nodes i and j belong to the same component if, and only if, there exists a circuit which contains i and j. The network is partitioned into R strongly connected components $X_1, \ldots, X_R$ such that $d_{ij} = \infty$ for all $i \epsilon X_t$, $j \epsilon X_s$ and $t > s$. It is shown that

(i) $d_{ij}^k = \infty$ for all $i \epsilon X_t$, $j \epsilon X_s$, $t > s$, $k = 0, \ldots N$

(ii) $d_{ij}^k = d_{ij}^{k-1}$

(a) for all $k \epsilon X_s$, $i \epsilon X_t$, $t > s$ or

(b) for all $k \epsilon X_s$, $j \epsilon X_t$, $s > t$.

The algorithm, based on these observations, computes the shortest path in N iterations.

Jensen [30] gives the optimum network partitioning in directed

acyclic networks. The dynamic programming algorithm calculates the optimum feasible partitioning by finding the nested set of restricted sets corresponding to the partition, where in a graph $G = (N,A)$ the node set $X \subset N$ is called the restricted set if for all $i \in X$, $j \in \bar{X}$, the $j \not\succ i$ (j does not precede i).

D. Minimal Cost Flow Algorithms

There are three basic approaches for solving the minimum cost network flow problems: primal algorithms, dual algorithms and primal-dual algorithms.

The primal algorithms start with a feasible maximal flow of f^* units, and try to improve the cost of sending f^* units from source to sink by detecting a negative cycle in the marginal cost network and circulating as much flow as possible around this negative cycle. The algorithms terminate when there exists no negative cycle in the marginal cost network with respect to the given feasible flow f^*.

The dual algorithms attempt to build up the optimal flow f^* by adding one unit of flow to the network over the shortest path (flow augmenting path) obtained on the marginal cost network. The advantage of dual algorithms over the primal algorithms is the relaxation of the need of maximal flow to start with.

An example of a primal-dual algorithm is the out-of-kilter algorithm proposed by Ford and Fulkerson [16]. The algorithm starts with any given flow and tries to maintain the primal and dual feasibility towards the optimal solution. The algorithm contains two phases: the flow increase and dual variable change. For each arc on the network a

kilter number is defined. The algorithm tries to reduce the kilter number of each arc while maintaining the primal feasibility. Any arc (i,j) which satisfies the Kuhn-Tucker conditions with respect to a given flow is said to be "in kilter" and the kilter number of such arc is said to be zero.

The direct search for negative cycles, by building the negative partial sums, is given by Florian and Robert [13]. Other primal algorithms utilize the shortest path algorithms to detect the negative cycles. Since the shortest path algorithms are not designed primarily for detecting the negative cycles, their efficiency is questionable. As Bennington [3] states, "if the relative efficiency of the search for negative cycles is not improved, it would seem that the primal method is in fact doomed".

Unfortunately, we cannot be too optimistic about Florian and Robert's [13] direct search for negative cycles. The algorithm is based on the property of negative partial sums of finite sequences. Shea [42] shows that this algorithm can be a poor choice for locating negative cycles. Depending upon the number of arcs in the negative cycle and its location, the algorithm might require far more computer time than any other competing shortest path algorithms used for detecting the negative cycles.

Edmonds and Karp [11] give algorithm for minimal cost flow problems where at each step the flow augmenting path is found on a marginal cost network in which the arc weighs are nonnegative. At each flow augmentation, the marginal cost network has nonnegative arc weights. Thus Dijkstra's algorithm can effectively be used to obtain the flow augmenting path in $0(n^2)$ operations.

CHAPTER III

LOCATION OF MIN-CUT BY DECOMPOSITION

A network is a collection of nodes and ordered pairs of nodes designed by G = (N,A), where N is the node-set and A is the arc-set of the network. If the ordered pairs (i,j) have assigned direction, then the network is called a directed network. Otherwise, it is called an undirected (symmetric) network.

Let P = {1, (1,2), 2, (2,3), . . ., k-1, (k-1,k), k} be a sequence of nodes and arcs. This sequence P is called a path if all the arcs have the same sense of direction. Otherwise, the sequence P is called a chain. If the initial and the final nodes of a path (chain) coincide then the resulting figure is called a cycle (circuit).

There are two special nodes in a network. One is called the source, denoted by s, and the other one is called the sink, denoted by t.

With every arc $(i,j) \in A$ of G we associate a positive integer u_{ij}, called the capacity of the arc and a real (or integer) number c_{ij} (or d_{ij}) the cost (length or weight) of the arc. A set of nonnegative integers f_{ij} is called a flow in a network if they satisfy the flow conservation equations

$$\sum_i f_{ij} - \sum_k f_{jk} = \begin{cases} -f & \text{if } j = s \\ 0 & \text{if } j \neq s, t \\ f & \text{if } j = t \end{cases}$$

$$0 \leq f_{ij} \leq u_{ij} \quad \text{for all } i,j$$

where, f is a nonnegative number called the value of the flow.

A cut is denoted by $(X,\bar{X})$, where $X \subset N$ is a subset of nodes of the network and $\bar{X}$ contains the remaining nodes of N, i.e., $\bar{X} = N - X$. Therefore, a cut $(X,\bar{X})$ denotes the subset of arcs (i,j) of A such that $(i,j) \epsilon (X,\bar{X})$ if $i \epsilon X$, $j \epsilon \bar{X}$ or $j \epsilon X$, $i \epsilon \bar{X}$. The capacity of a cut $(X,\bar{X})$, denoted by $c(X,\bar{X})$ is

$$c(X,\bar{X}) = \sum_{\substack{i \epsilon X \\ j \epsilon \bar{X}}} u_{ij}.$$

A cut separating s from t with minimum capacity is called a minimum cut. Consider a network G = (N,A) with a feasible flow f. Let G' = (N,A) be another network obtained from G such that, G' contains the same nodes as network G. The arcs of G' is defined as:

$$(i,j) \epsilon A' \text{ if } (i,j) \epsilon A \text{ and } f_{ij} < u_{ij}$$

with $u'_{ij} = u_{ij} - f_{ij}$,

$$(j,i) \epsilon A' \text{ if } (i,j) \epsilon A \text{ and } f_{ij} > 0.$$

with $u'_{ji} = f_{ij}$, where f_{ij} is the flow on arc (i,j) with respect to the flow f in network G. The network G' obtained in this manner is called the marginal network. If a path from s to t exists in G' with respect to flow f, then we can increase the flow in G by increasing the flow along this path. This is the basic idea of the flow augmenting algorithms.

The following algorithm given by Edmonds and Karp [11] determines the maximum flow by flow augmentations.

Max-Flow Algorithm [11]:

Start with any feasible flow f^o in G, say $f_{ij} = 0$. From G construct G' as described above. If a path P exists in G', from node s to node t,construct a new feasible flow $f' = f + \Delta$ by sending as much additional flow as is possible along the path P, where $\Delta = \min \{u'_{ij} : (i,j) \in P\}$. For this new feasible flow construct G' network and repeat the procedure again. At each flow augmentation choose a path with the minimum number of arcs in it. If no path exists in G' with respect to flow f', then stop. f' is the maximal flow. The set of nodes X that can be reached from source s and the remaining nodes $\bar{X} = N - X$ constitutes a minimum cut for the network G = (N,A).

Corollary:

A cut $(X,\bar{X})$ in a network G = (N,A) is minimal if and only if every maximal flow f saturates all arcs of $(X,\bar{X})$, whereas all arcs of $(\bar{X},X)$ are flowless with respect to f.

Theorem 1 (27): Let G = (N,A) be a network with two minimum cuts $(X,\bar{X})$ and $(Y,\bar{Y})$ where $X \cup \bar{X} = Y \cup \bar{Y} = N$ and $X \cap \bar{X} = Y \cap \bar{Y} = \phi$ and $X \neq Y$. Then, $(X \cup Y, \overline{X \cup Y})$ and $(X \cap Y, \overline{X \cap Y})$ are also minimum cuts.

Proof: See T. C. Hu [27].

Definition: Two cuts $(X,\bar{X})$ and $(Y,\bar{Y})$ are said to cross each other if and only if each of the four sets $X \cap Y$, $\bar{X} \cap Y$, $X \cap \bar{Y}$, $\bar{X} \cap \bar{Y}$ are nonempty.

The following three lemmas apply only to undirected networks.

LEMMA 1 (27). Let $(X,\bar{X})$ be a minimum cut separating node $i \in X$

and some other node, and let e and k be any two nodes contained in $\bar{X}$. Then, there exists a minimum cut $(Z,\bar{Z})$ separating nodes e and k such that $(Z,\bar{Z})$ and $(X,\bar{X})$ do not cross each other.

Proof: See T. C. Hu 27 .

LEMMA 2 (27). Let $(X,\bar{X})$ be a minimum cut separating node i and some other nodes, and let e be any node that belongs to $\bar{X}$. Then, there exists a minimum cut $(Z,\bar{Z})$ separating nodes i and e such that $(X,\bar{X})$ and $(Z,\bar{Z})$ do not cross each other.

Proof: See T. C. Hu [27].

LEMMA 3 (27). Let $f_{ab} = c(X,\bar{X})$ and i and j be any two nodes with $i\epsilon X$ and $j\epsilon\bar{X}$. Then, there exists a minimum cut $(Z, \bar{Z})$ with $c(Z, \bar{Z}) = f_{ij}$ such that $(Z,\bar{Z})$ does not cross $(X,\bar{X})$.

Proof: See T. C. Hu [27].

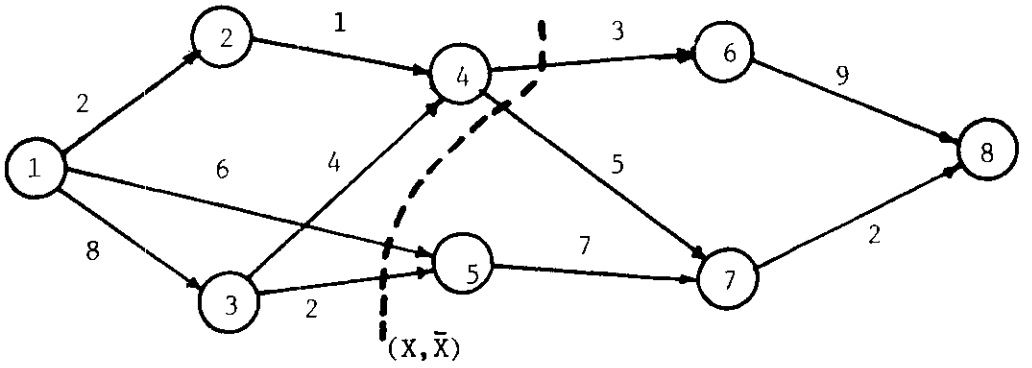
Developed Theory

This section strictly deals with the single commodity, directed networks.

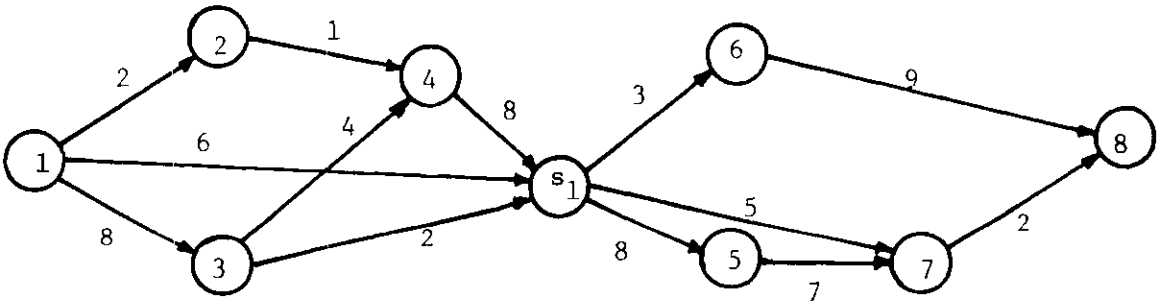
Definition: A cut $(X,\bar{X})$ is called a "min-source cut" between nodes i and j of $G = (N,A)$ if $c(X,\bar{X}) \leq c(Z,\bar{Z})$ for all $(Z,\bar{Z})$ such that $X \neq Z$, s, $i\epsilon X$, s, $i\epsilon Z$, j, $t\epsilon\bar{X}$ and j, $t\epsilon\bar{Z}$.

Property: When i = s and j = t, the min-source-cut $(X,\bar{X})$ becomes the actual min-cut for $G = (N,A)$.

Consider the network G = (N,A) given below.



Suppose the node-set N of G is divided into two subsets X and $\bar{X}$ such that $X \cap \bar{X} = \phi$ and $X \cup \bar{X} = N$. Suppose, also that the arcs on $(X,\bar{X})$ are collected into a super source s_1, first, and then reflected into $\bar{X}$ as shown below.



The flow capacities of the arcs in the form (i,s_1) or (s_1, j) are defined as:

$$u_{i,s_1} = \Sigma\, u_{ij} \quad \text{for each } i \in X \text{ and all } j \in \bar{X}.$$

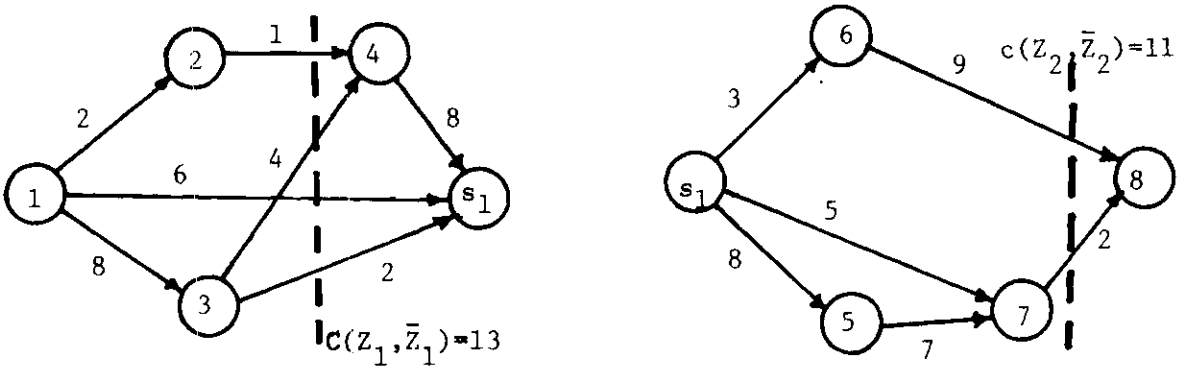
Similarly,

$$u_{s_1,j} = \Sigma\, u_{ij} \quad \text{for each } j \in \bar{X} \text{ and all } i \in X.$$

Let this new network be defined as $G' = (X \cup s_1 \cup \bar{X}, A')$. Define two new

subnetworks $G_1 = (N_1, A_1)$ and $G_2 = (N_2, A_2)$ where $N_1 = X \cup s_1$, $N_2 = \bar{X} \cup s_1$, $(i,j) \in A_1$ if and only if $(i,j) \in A'$ and $i \in N_1$. Similarly $(i,j) \in A_2$ if and only if $(i,j) \in A'$ and $i \in N_2$, $j \in N_2$.

If we apply the Ford and Fulkerson's [16] labeling algorithm on G_1 and G_2 separately, we obtain two minimum cuts $(Z_1, \bar{Z}_1)$ and $(Z_2, \bar{Z}_2)$, respectively. These two cuts are shown on the example network below.



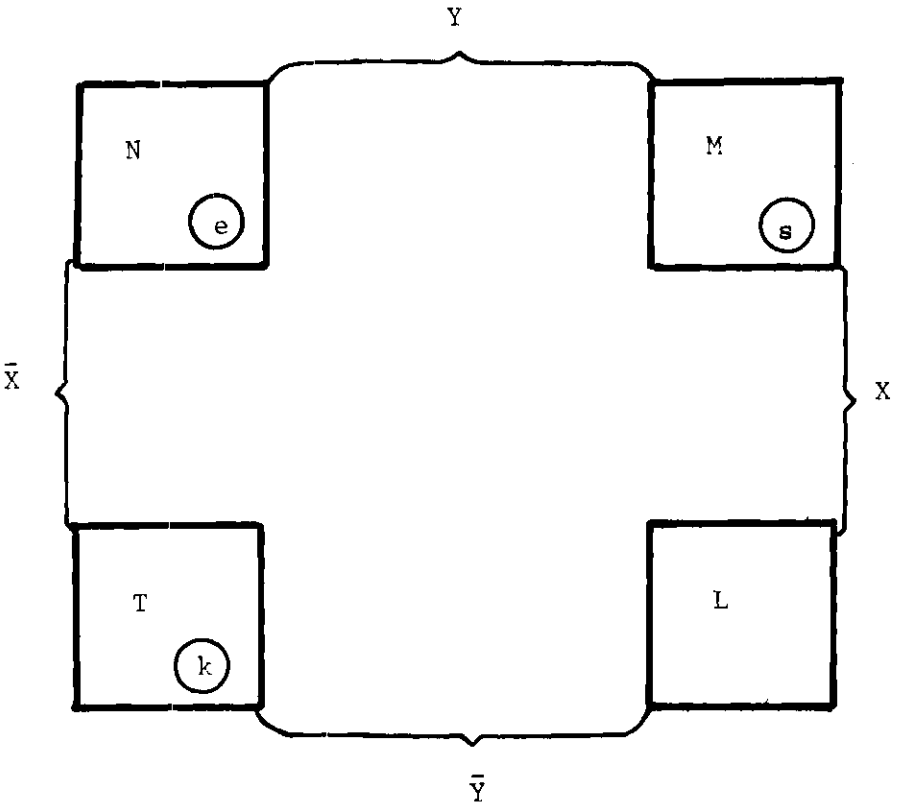
After studying the locations of $(Z_1, \bar{Z}_1)$, $(Z_2, \bar{Z}_2)$, $(X, \bar{X})$ and the actual min-cut of the original network $G = (N,A)$ we can prove the following three lemmas which are inspired from T. C. Hu's [27] three lemmas. The essential difference is the fact that the proposed lemmas apply to the directed networks as well.

LEMMA 4. Let $(X, \bar{X})$ be a min-source cut separating $s \in X$ and some other node. Let e and k be any two nodes contained in $\bar{X}$. There exists a min-source cut $(Z, \bar{Z})$ separating e and k $(s \in Z)$ such that $(Z, \bar{Z})$ and $(X, \bar{X})$ do not cross each other.

Proof: Assume that there is a min-source cut $(Y, \bar{Y})$ separating e and k $(s \in Y)$ which does cross $(X, \bar{X})$. Let $X \cap Y = M$ $\quad X \cap \bar{Y} = L$

$\bar{X} \cap Y = N$ $\quad \bar{X} \cap \bar{Y} = T$

as shown in the following figure.


Y
N
e
M
s
$\bar{X}$
X
T
k
L
$\bar{Y}$

Since $s \epsilon Y$ and $s \epsilon X$, s will always be in $M = X \cap Y$. With the same line of argument, $e \epsilon Y$ and $e \epsilon \bar{X}$ therefore, $e \epsilon N = Y \cap \bar{X}$. Since $k \epsilon \bar{Y}$ and $k \epsilon \bar{X}$, k will always be in $T = \bar{X} \cap \bar{Y}$. Therefore, the above figure is the only possible case.

Since $(X,\bar{X})$ is the min-source cut, we have

$$C(M,N) + C(L,N) + C(M,T) + C(L,T) \leq C(M,N) + C(M,T) + C(M,L).$$

Since $C(L,N) \geq 0$,

$$C(L,T) \leq C(M,L). \tag{1}$$

Since $$C(N,T) + C(M,T) = C(N,T) + C(M,T) \tag{2}$$

and $$0 \leq C(N,L), \tag{3}$$

adding both sides of (1), (2) and (3) we have

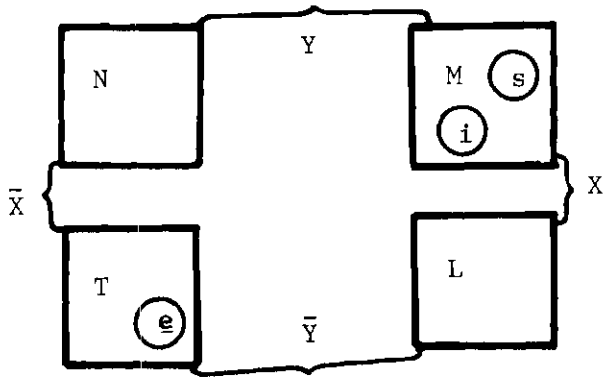
$$C(L,T) + C(N,T) + C(M,T) \leq C(M,L) + C(N,L) + C(N,T) + C(M,T). \tag{4}$$

The left-hand side of (4) is the value of a source cut (NULUM, T) which separates e and k ($s \epsilon$ NULUM) that does not cross $(X,\bar{X})$. The right-hand side of (4) is the min-source cut $(Z,\bar{Z})$.

LEMMA 5. Let $(X,\bar{X})$ be a min-source cut separating node i and some other node, and let e be any node that belongs to $\bar{X}$. Then there exists a min-source cut $(Z,\bar{Z})$ separating i and e ($s \epsilon Z$) such that $(X,\bar{X})$ and $(Z,\bar{Z})$ do not cross each other.

Proof: Assume that there is a min-source cut $(Y,\bar{Y})$ separating i and e that does cross $(X,\bar{X})$ From the construction of the lemma, we have $s \epsilon X$, $s \epsilon Y$, $i \epsilon Y$ and $e \epsilon \bar{X}$, $e \epsilon \bar{Y}$. Therefore the following figure is the only case

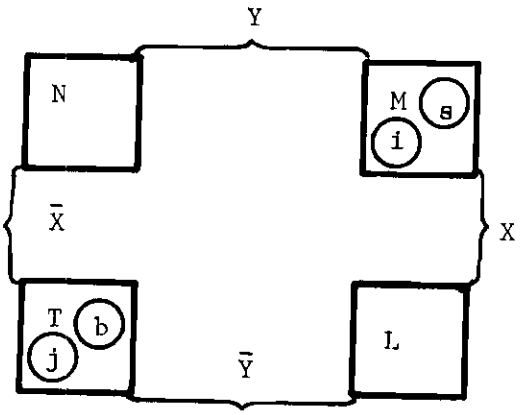
for this lemma.



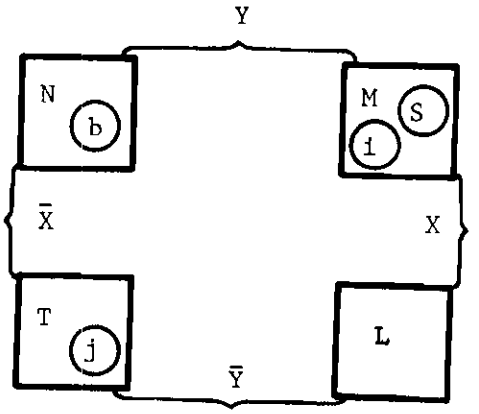
As in lemma 4, with the same line of argument we have (4). Therefore, (NULLM, T) is the required min-source cut between i and e that does not cross $(X,\bar{X})$.

LEMMA 6. Let $(X,\bar{X})$ be a min-source cut between s and b. Let i and j be any two nodes with $i\epsilon X$ and $j\epsilon\bar{X}$. Then there exists a min-source cut $(Z,\bar{Z})$ separating i and j such that $(X,\bar{X})$ and $(Z,\bar{Z})$ do not cross each other.

Proof: Assume that there is a min-source cut $(Y,\bar{Y})$ separating nodes i and j that does cross $(X,\bar{X})$. We have $s\epsilon X$, $s\epsilon Y$, hence $s\epsilon M$. We also have $i\epsilon X$, $i\epsilon Y$, hence $i\epsilon M$. With the same line of argument $j\epsilon T$. There are two possibilities for b: it is either in N, or it is in T. These two cases are shown in figures below.



CASE I



CASE II

CASE I. $b \in T$. Since $(X,\bar{X})$ is a min-source cut we have $C(M,N) + C(M,T) + C(L,N) + C(L,T) \leq C(M,N) + C(M,T) + C(M,L)$.

Since
$$C(L,N) \geq 0 \tag{5}$$
$$C(L,T) \leq C(M,L). \tag{6}$$
Since
$$C(N,T) + C(M,T) = C(N,T) + C(M,T), \tag{7}$$
and
$$0 \leq C(N,L), \tag{8}$$
adding both sides of (6), (7) and (8) we have
$$C(L,T) + C(N,T) + C(M,T) \leq C(M,L) + C(N,L) + C(M,T) + C(N,T). \tag{9}$$
But the left-hand side of (9) is the source cut $(L \cup N \cup M, T)$ which separates i from node j (with $s \in L \cup N \cup M$) that does not cross $(X,\bar{X})$. The right-hand side of (9) is the min-source cut $(Y,\bar{Y})$. Hence, the cut $(M \cup L \cup N, T)$ is the required min-source cut $(Z,\bar{Z})$.

CASE II. $b \in N$. With exactly the same reasoning we have (9). Therefore the required min-source cut is $(M \cup N \cup L, T)$ that does not cross $(X,\bar{X})$.

We can now state two lemmas which will be used in partitioning of a network into two subnetworks.

LEMMA 7. Let a network $G = (N,A)$ be partitioned into two subnetworks by the cut $(X,\bar{X})$ such that $X \cap \bar{X} = \phi$ and $X \cup \bar{X} = N$, $s \in X$ and $t \in \bar{X}$ where s is the source node and t is the sink node. Let all the arcs (i,j) such that $i \in X$, $j \in \bar{X}$ be collected to an intermediate node s' and arcs leaving X into $\bar{X}$ enters into s' first and then enters to $\bar{X}$, i.e., $C(X,s') = C(s', \bar{X})$

Let $G_1 = (XUs', A_1)$ where $(i,j)\epsilon A_1$ if $i\epsilon XUs'$ and $j\epsilon XUs'$ and (i,j) exists, and let $G_2 = (\bar{X}Us', A_2)$ where $(k,l)\epsilon A_2$ if $k\epsilon \bar{X}Us'$, $l\epsilon XUs'$ and (k,l) exists.

Suppose that $(Z_1,\bar{Z}_1)$ is the min-cut obtained in G_1 between nodes s and s' and $(Z_2,\bar{Z}_2)$ is the min-cut obtained in G_2 between nodes s' and t. Then, we can state:

(a) $\max\left\{c(Z_1,\bar{Z}_1),\ c(Z_2,\bar{Z}_2)\right\} \leq c(X,\bar{X})$

(b) if $c(Z_1,Z_1) = c(X,\bar{X})$ then either both cuts are the min-cuts for G or the min-cut for G lies on the right of $(X,\bar{X})$.

(c) If $c(Z_2,\bar{Z}_2) = c(X,\bar{X})$ then either both cuts are min-cuts to G or the actual min-cut to G lies on the left of $(X,\bar{X})$.

(d) If $c(X,\bar{X}) > c(Z_1,\bar{Z}_1)$, $c(Z_2,\bar{Z}_2)$ then the minimum cut is either the minimum of $c(Z_1,\bar{Z}_1)$ and $c(Z_2,\bar{Z}_2)$ or it lies in between $(Z_1,\bar{Z}_1)$ and $(Z_2,\bar{Z}_2)$ and crosses $(X,\bar{X})$.

Proof: (a) If $c(Z_1,\bar{Z}_1) > c(X,\bar{X})$, then we would have chosen $(X,\bar{X})$ as the min-source cut between nodes s and s'. Hence $c(Z_1,\bar{Z}_1) \leq c(X,\bar{X})$. With the same line of argument, if $c(Z_2,\bar{Z}_2) > c(X,\bar{X})$ then we would have chosen $(X,\bar{X})$ as the min-source cut between nodes s' and t. Therefore, $c(Z_2,\bar{Z}_2) \leq c(X,\bar{X})$.

(b) We assumed that $c(Z_1,\bar{Z}_1) = c(X,\bar{X})$. There exists two possibilities for $c(Z_2,\bar{Z}_2)$.

(i) $c(Z_2,\bar{Z}_2) = c(X,\bar{X})$.

Let $(Y,\bar{Y})$ be the actual min-cut for G and cross $(X,\bar{X})$. Suppose $s'\epsilon\bar{Y}$. We have $s\epsilon Y$ and other node being $t\epsilon\bar{Y}$. Hence, $(Y,\bar{Y})$ is the min-cut that

separates s and t. Let e = s' and k = t. From lemma 4 there exists a min-source cut $(X,\bar{X})$ between $s' = e$ and t = k such that $(Y,\bar{Y})$ and $(X,\bar{X})$ do not cross each other. Since $(X,\bar{X})$ is the min-source cut between s' and t, therefore, $(Y,\bar{Y})$ cannot cross $(X,\bar{X})$.

Now, suppose $s' \epsilon Y$, and let again $(Y,\bar{Y})$ be the min-cut between s and t, that crosses $(X,\bar{X})$. Let b = t, i = s' and j = t. From lemma 6, the min-source cut $(X,\bar{X})$ between i = s' and j = t cannot cross $(Y,\bar{Y})$. Therefore, if $(Y,\bar{Y})$ cannot cross $(X,\bar{X})$, then $(Y,\bar{Y})$ must lie entirely in G_1 or G_2. In any case the min-source cuts obtained for each subnetwork will be the actual min-cut to the original problem as well as $(X,\bar{X})$.

(ii) $c(Z_2,\bar{Z}_2) < c(X,\bar{X}) = c(Z_1,\bar{Z}_1)$.

Suppose the actual min-cut for G is $(Y,\bar{Y})$ and crosses $(X,\bar{X})$. Let $s' \epsilon \bar{Y}$, e = s', i = s and t be the other node. From lemma 5 the min-cut $(Y,\bar{Y})$ between s and t does not cross the min-source cut $(X,\bar{X})$ between s and s'.

Let $s' \epsilon Y$, b = s', i = s, j = t then from lemma 6, the min-source cut $(X,\bar{X})$ between s and s' cannot cross the min-cut $(Y,\bar{Y})$.

Therefore the min-cut $(Y,\bar{Y})$ is on the right of $(X,\bar{X})$, i.e., it lies entirely in G_2. Hence $(Z_2,\bar{Z}_2)$ is the required min-cut.

(c) $c(Z_2,\bar{Z}_2) = c(X,\bar{X})$.

We have again two possibilities for $c(Z_1,\bar{Z}_1)$.

(i) $c(Z_1,\bar{Z}_1) = c(X,\bar{X}) = c(Z_2,\bar{Z}_2)$. This is nothing but case (b - i).

(ii) $c(Z_1,\bar{Z}_1) < c(X,\bar{X}) = c(Z_2,\bar{Z}_2)$. Let the actual min-cut

Page missing from thesis

for G be $(Y,\bar{Y})$ and let it cross $(X,\bar{X})$.

If $s' \epsilon Y$, let $(X,\bar{X})$ be the min-source between s and s'. Let b = t. Then from lemma 6, there exists a min-cut $(Y,\bar{Y})$ between s and t such that $(X,\bar{X})$ and $(Y,\bar{Y})$ do not cross each other. If $s' \epsilon \bar{Y}$, again by letting $e = s'$, and $t = k$, the min-source cut $(X,\bar{X})$ separating s' and t cannot cross the min-cut $(Y,\bar{Y})$ between s and t. Thus, $(Z_1,\bar{Z}_1)$ is the min-cut for G.

(d) $c(X,\bar{X}) > c(Z_1,\bar{Z}_1)$, $c(Z_2,\bar{Z}_2)$.

Let the actual min-cut between s and t be $(Y,\bar{Y})$ and suppose it crosses $(Z_1,\bar{Z}_1)$.

If $s' \epsilon \bar{Y}$, then from lemma 5 with i = s and the other node being t and $e = s'$, the min-cut $(Y,\bar{Y})$ between s and t cannot cross the min-source cut $(Z_1,\bar{Z}_1)$ between s and s'.

If $s' \epsilon Y$, then let $(Z_1,\bar{Z}_1)$ be the min-source cut between $i = s$ and s'. Let e = t. From lemma 5, the min-cut between s and t does not cross $(Z_1,\bar{Z}_1)$.

Now, we show that the actual min-cut for G cannot cross $(Z_2,\bar{Z}_2)$ either. Suppose the actual cut $(Y,\bar{Y})$ does cross $(Z_2,\bar{Z}_2)$. If $s' \epsilon Y$, let $b = t$, $s' = i$ and $j = t$. From lemma 6 there exists a min-source cut $(Z_2,\bar{Z}_2)$ between s' and t such that the min-cut $(Y,\bar{Y})$ between s and t does not cross $(Z_2,\bar{Z}_2)$. If $s' \epsilon \bar{Y}$, let $e = s'$, k = t and the other node being t. From lemma 4, the min-source cut $(Z_2,\bar{Z}_2)$ between the nodes s' and t cannot cross $(Y,\bar{Y})$.

Summarizing the above proof, we have:

$c(Z_1,\bar{Z}_1) < c(Z_2,\bar{Z}_2) < c(X,\bar{X})$, then the actual min-cut $(Y,\bar{Y})$ for G cannot cross the cuts $(Z_1,\bar{Z}_1)$ and $(Z_2,\bar{Z}_2)$. Since $c(Z_1,\bar{Z}_1) < c(Z_2,\bar{Z}_2)$,

the min-cut for G is either $(Z_1,\bar{Z}_1)$ or it lies between $(Z_1,\bar{Z}_1)$ and $(Z_2,\bar{Z}_2)$ and it <u>crosses</u> $(X,\bar{X})$, (if not, then we would have picked it as $(Z_j,\bar{Z}_j)$).

If $c(Z_2,\bar{Z}_2) < c(Z_1,\bar{Z}_1) < c(X,\bar{X})$, then either $(Z_2,\bar{Z}_2)$ is the min-cut for G or it lies between $(Z_1,\bar{Z}_1)$ and $(Z_2,\bar{Z}_2)$, and it crosses $(X,\bar{X})$. Q.E.D.

Results:

I. If $c(Z_1,\bar{Z}_1) = c(X,\bar{X}) = c(Z_2,\bar{Z}_2)$, then each of those cuts is a min-cut for G.

II. If $c(Z_1,\bar{Z}_1) = c(X,\bar{X}) > c(Z_2,\bar{Z}_2)$, then $(Z_2,\bar{Z}_2)$ is the min-cut for G.

III. If $c(Z_2,\bar{Z}_2) = c(X,\bar{X}) > c(Z_1,\bar{Z}_1)$, then $(Z_1,\bar{Z}_1)$ is the min-cut for G.

IV. If $c(Z_1,\bar{Z}_1)$ and $c(Z_2,\bar{Z}_2) < c(X,\bar{X})$, then the min-cut for G is either

(a) $\min\left\{c(Z_1,\bar{Z}_1),\ c(Z_2,\bar{Z}_2)\right\}$ or

(b) it lies entirely between $(Z_1,\bar{Z}_1)$ and $(Z_2,\bar{Z}_2)$ and it <u>crosses</u> $(X,\bar{X})$.

<u>Redistributing the Nodes</u>

Suppose we decomposed the network G = (N,A) into two subnetworks, $G_1 = (X \cup s',\ A_1)$ and $G_2 = (\bar{X} \cup s',\ A_2)$ as explained earlier. Furthermore, suppose that there is an $(s',\ \bar{X})$ arc in the minimum cut obtained for the second subnetwork G_2. By bringing the node $k \in X$ to the set $\bar{X}$ we are trying to find out new partitioning $(X',\bar{X}')$ and $(Z_2',\bar{Z}_2')$ such that they will

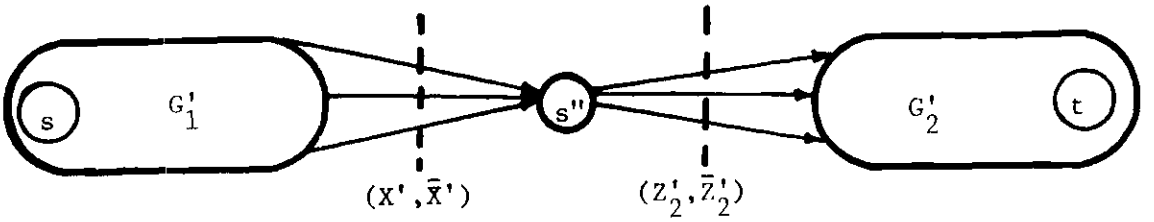
be disjoint.

Everytime we bring node(s) from G_1 to G_2 we perform a max-flow operation to see whether the new min-cut for G_2' is disjoint from $(X', \bar{X}')$ or not. After finding such disjoint cuts $(X',\bar{X}')$ and $(Z_2',\bar{Z}_2')$; all we need to do is to transfer the nodes on the left of $(Z_2',\bar{Z}_2')$ to the first subnetwork (i.e., make $(Z_2',\bar{Z}_2')$ a new $(X',\bar{X}')$) and solve the max-flow problem on this new first subnetwork G_1'. By so doing, we are assuring that $c(X',\bar{X}') = c(Z_2',\bar{Z}_2')$. From part II of the results of lemma 7, if $c(Z_1',\bar{Z}_1')$ found for G_1' is less than the other two, then it is the actual cut for the original network G. If $c(Z_1',\bar{Z}_1') = c(X',\bar{X}')$ then from result I, $(Z_1',\bar{Z}_1')$ and $(X',\bar{X}')$ are the min-cuts for G.

Utilizing these observations, we can prove the following lemma:

LEMMA 8. After the redistributing operation described above, the min-cut obtained is the actual min-cut for the overall network G.

Proof: After bringing the nodes into G_2, and finding the disjoint min-cut $(Z_2',\bar{Z}_2')$ from $(X',\bar{X}')$, let $(X',\bar{X}') = (Z_2',\bar{Z}_2')$. We now have the following:

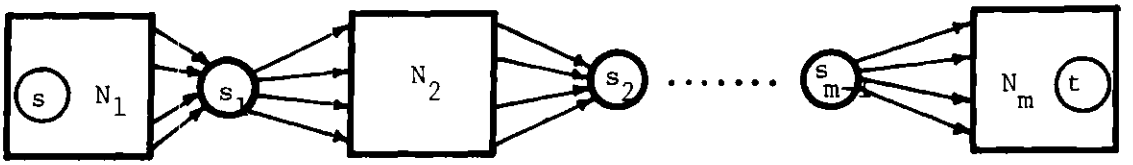


Let $e = s''$, $k = t$. From lemma 4 the min-source cut $(X',\bar{X}') = (Z_2,\bar{Z}_2)$ cannot cross the min-cut $(Y,\bar{Y})$ between the nodes s and t. Therefore, the min-cut $(Y,\bar{Y})$ entirely lies in the network $G_1' = (X' \cup s'', A'_1)$.

If the min-cut $(Z_1', \bar{Z}_1')$ obtained from G_1' is different from $(X', \bar{X}')$ and if $c(Z_1', \bar{Z}_1') < c(X', \bar{X}')$, then $(Z_1', \bar{Z}_1')$ is the actual min-cut for G. If $c(Z_1', \bar{Z}_1') = c(X', \bar{X}')$ and they are different, then both are min-cuts for G. If they are the same, then $(X', \bar{X}')$ is the min-cut for G.

A Decomposition Algorithm for Locating Min-Cut(s)

We will now give the proposed decomposition algorithm for locating a min-cut. Let the node set N of a network G = (N,A) be divided into m disjoint subsets $N_1, N_2, \ldots, N_m$ such that $N_i \cap N_j = \emptyset$ and $\bigcup_{i=1}^{m} N_i = N$ for all i and j. Let the arcs on each cut be collected into the intermediate artificial nodes $s_1, s_2, \ldots, s_{m-1}$, as shown below.



Let $G_1 = (N_1 \cup s_1, A_1)$, $G_2 = (s_1 \cup N_2 \cup s_2, A_2) \ldots, G_m = (s_{m-1} \cup N_m, A_m)$ where A_j, j=1, . . . , m as defined before. Let those m arbitrary cuts be represented by $(X_i, \bar{X}_i)$ i=1, . . . m-1. Also, let $(Z_i, \bar{Z}_i)$ i=1, . . . , m be the min-cuts for the networks $G_1, G_2, \ldots, G_m$, respectively.

The following algorithm fans from one end of the network.

Algorithm A. Locating the min-cut by decomposition.

Step 0. Let k=1, $(Y_k, \bar{Y}_k) = (Z_k, \bar{Z}_k)$.

Step 1. If k=m, stop. Minimum cut $(Y_k, \bar{Y}_k)$ is at hand. Otherwise,

(i) Assume the overall network consists of only $\bigcup_{i=1}^{k+1} G_i$.

Suppose that $(X_k, \bar{X}_k)$ divides this network into two subnetworks. We have

at hand, $c(Y_k,\bar{Y}_k)$ the min-cut for $\bigcup_{i=1}^{k} G_i$, $c(X_k,\bar{X}_k)$ the cut that divides the network into two and $c(Z_{k+1}, \bar{Z}_{k+1})$ the minimum cut for the subnetwork G_{k+1}.

Apply lemma 7 to locate the actual min-cut for the network $\bigcup_{i=1}^{k+1} G_i$. Let this cut be $(Y_{k+1},\bar{Y}_{k+1})$ and let k = k+1. Go to the beginning of Step 1.

Theorem 3. Algorithm A converges to the minimum cut in exactly m-1 steps.

Proof: The algorithm starts with $G = G_1 \cup G_2$ and assumes $(X_1,\bar{X}_1)$ divides this network into G_1 and G_2. We have at hand $(Z_1,\bar{Z}_1)$ corresponding to G_1 and $(Z_2,\bar{Z}_2)$ corresponding to G_2. By applying lemma 7 we will locate the min-cut $(Y_1,\bar{Y}_1)$ for the network $G_1 \cup G_2$. In the next step the algorithm assumes that the overall network consists of $G_1 \cup G_2 \cup G_3$. And it further assumes that $(X_2,\bar{X}_2)$ divides this network into two subnetworks $G_1 \cup G_2$ and G_3, respectively. From the previous step we have $(Y_1,\bar{Y}_1)$, the min-cut for $G_1 \cup G_2$, and we have $(Z_3,\bar{Z}_3)$, the min-cut for G_3. Hence lemma 7 is applicable. The result of this step will give the minimum cut $(Y_2,\bar{Y}_2)$ for the network $G_1 \cup G_2 \cup G_3$.

In general, at the k^{th} iteration, we are considering the network $G_1 \cup G_2 \cup \ldots \cup G_k \cup G_{k+1}$. Assuming that $(X_k,\bar{X}_k)$ divides this network into $G_1 \cup G_2 \cup \ldots \cup G_k$ and G_{k+1}, respectively, we have at hand $(Y_{k-1}, \bar{Y}_{k-1})$, the minimum cut for the network $G_1 \cup \ldots \cup G_k$, and $(Z_k, \bar{Z}_k)$ the min-cut for the network G_{k+1}. The conclusion of lemma 7 will produce the min-cut $(Y_k,\bar{Y}_k)$ for the network $G_1 \cup G_2 \cup \ldots \cup G_{k+1}$. When k = m-1 we are considering $G = G_1 \cup G_2 \cup \ldots \cup G_{k+1}$. Assume that

$(X_{m-1}, \bar{X}_{m-1})$ divides this network into two subnetworks $G_1 \cup G_2 \cup \ldots \cup G_{m-1}$ and G_m, respectively. From the $(m-2)^{nd}$ iteration we have the min-cut $(Y_{m-2}, \bar{Y}_{m-2})$ for the network $G_1 \cup G_2 \cup \ldots \cup G_{m-1}$, and we know the min-cut $(Z_m, \bar{Z}_m)$ for the network G_m. Therefore, the application of lemma 7 will produce the min-cut $(Y_{m-1}, \bar{Y}_{m-1})$, which is the actual min-cut for the network $G = G_1 \cup G_2 \cup \ldots \cup G_m$, the original network. Q.E.D.

Note that the fanning process is arbitrary. We could as well fan out from the last subnetwork towards the first subnetwork. In this case we start with $G_m \cup G_{m-1}$ and assume that $(X_{m-1}, \bar{X}_{m-1})$ divides this network into two. Upon applying lemma 7 we locate the min-cut $(Y_1, \bar{Y}_1)$, the min-cut for $G_m \cup G_{m-1}$. In general, for the k^{th} step, consider $G_m \cup G_{m-1} \cup \ldots \cup G_{m-k}$. Suppose the cut $(X_{m-k}, \bar{X}_{m-k})$ divides this network into two subnetworks, G_{m-k} and $G_{m-k+1} \cup G_{m-k+2} \cup \ldots \cup G_m$, respectively. From the $(k-1)^{st}$ step we have the min-cut $(Y_{k-1}, \bar{Y}_{k-1})$, for the network $G_{m-k+1} \cup G_{m-k+2} \cup \ldots \cup G_m$. We know the min-cut $(Z_{m-k}, \bar{Z}_{m-k})$ for G_{m-k}. Therefore, lemma 7 is applicable. The minimum cut $(Y_k, \bar{Y}_k)$ obtained from this step will correspond to the min-cut for the network $G_m \cup G_{m-1} \cup \ldots \cup G_{m-k}$. For $k = m-1$, the minimum cut $(Y_k, \bar{Y}_k)$ obtained will correspond to the minimum cut for the overall network $G = G_m \cup G_{m-1} \cup \ldots \cup G_1$.

The following is another alternative for the fanning process. Suppose all $c(Z_i, \bar{Z}_i)$ $i=1 \ldots, m$ are computed. Obviously, the values of those min-cuts form k blocks such that within each block the min-cut values will form either increasing or decreasing sequences as shown below.

$$|c(Z_1,\bar{Z}_1) \geq c(Z_2,\bar{Z}_2) \geq \ldots \geq c(Z_i,\bar{Z}_i)| \geq |c(Z_{i+1},\bar{Z}_{i+1}) \geq \ldots \geq c(Z_j,\bar{Z}_j)| \geq$$

$$|c(Z_{j+1},\bar{Z}_{j+1}) \leq \ldots \leq c(Z_p,\bar{Z}_p)| \ldots |c(Z_q,\bar{Z}_q) \leq \ldots \leq c(Z_m,\bar{Z}_m)|$$

Let the blocks be named $S_1, S_2, \ldots, S_k$. The fanning process will be as follows:

1. Find the min-cut $(Y_i,\bar{Y}_i)$ for each block S_i, i=1, . . ., k, starting with the max $\{c(Z_j,\bar{Z}_j)$ where $(Z_j,\bar{Z}_j) \epsilon S_i$, and the $(Z_{j+1},\bar{Z}_{j+1}) \epsilon S_i$ $\left[(Z_{j-1},\bar{Z}_{j-1}) \epsilon S_i\right]$ if the block is a decreasing [increasing] sequence. Fan out in the other direction until the last subnetwork in that block is considered. The result will give us the min-cut $(Y_i,\bar{Y}_i)$ for the block S_i.

Obviously, $c(Y_i,\bar{Y}_i)$ will form k, new blocks similar to that which was explained earlier, where $k_1 < k$. Call these blocks S_1', S_2', . . ., S_{k_1}'. Again, find the min-cut $(Y_i',\bar{Y}_i')$ for each block S_i', i=1 . . ., k_1, as explained earlier. The new values of $c(Y_i',\bar{Y}_i')$ will form $k_2 < k_1$ new blocks, etc. Finally, we will only have one block, with all the cut capacities $c(Y_i^{(t)},\bar{Y}_i^{(t)})$ forming either a strictly increasing or a decreasing sequence. Choose max $\left\{c(Y_i^{(t)},\bar{Y}_i^{(t)})\right\}$ and expand in the other direction. The result of this last fanning will give the desired min-cut $(Y,\bar{Y})$ for the overall network G.

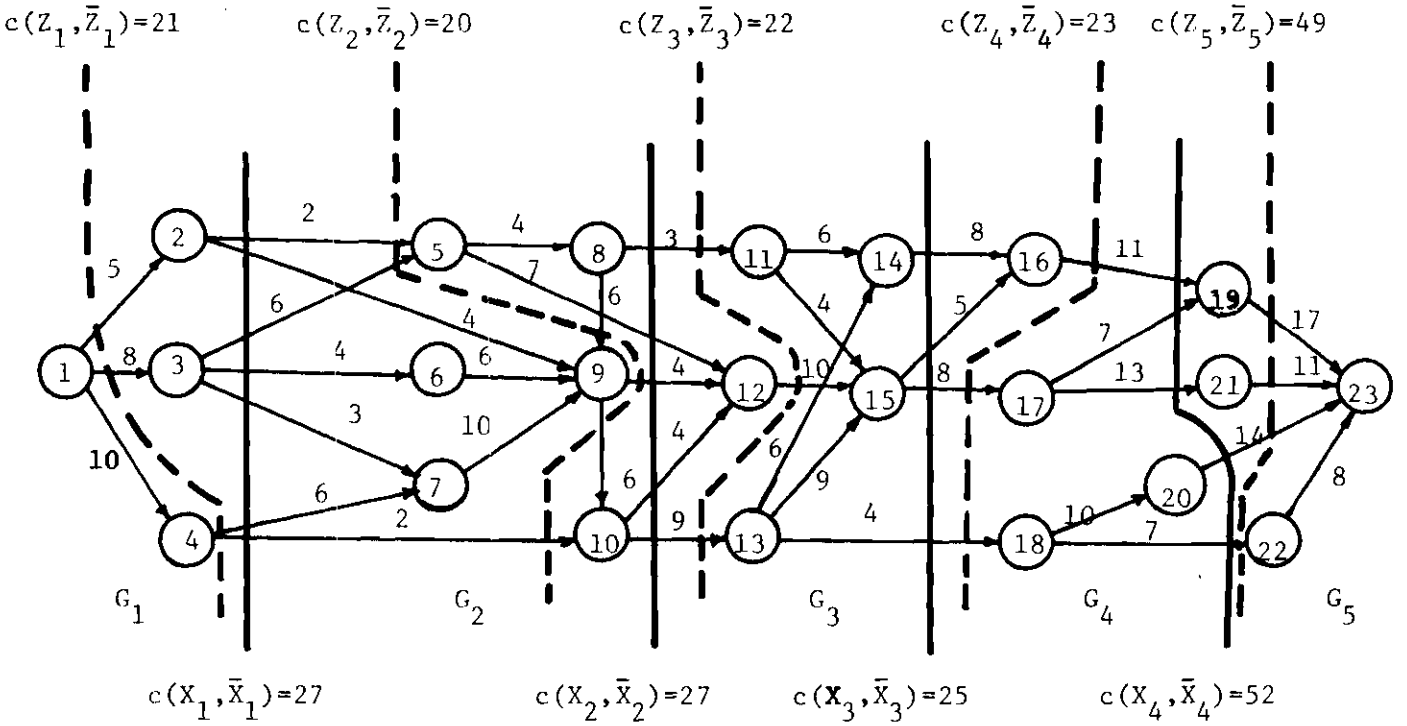
By adopting this fanning rule, we are making sure that at any step we are dealing with the minimum amount of nodes. In other words,

if lemma 7 -iv holds at every step and the actual min-cut for the two-partitioned network is $\min\{c(Z_i,\bar{Z}_i),\ c(Y_k,\bar{Y}_k)\}$, we want to be sure that we will use the minimum number of nodes to detect this fact.

The example problem will make this point clearer. The same problem is solved by using two different fanning procedures, fanning from the first subnetwork towards the last subnetwork and working with blocks. Working with blocks enabled us to consider the minimum number of nodes possible at each iteration.

Example

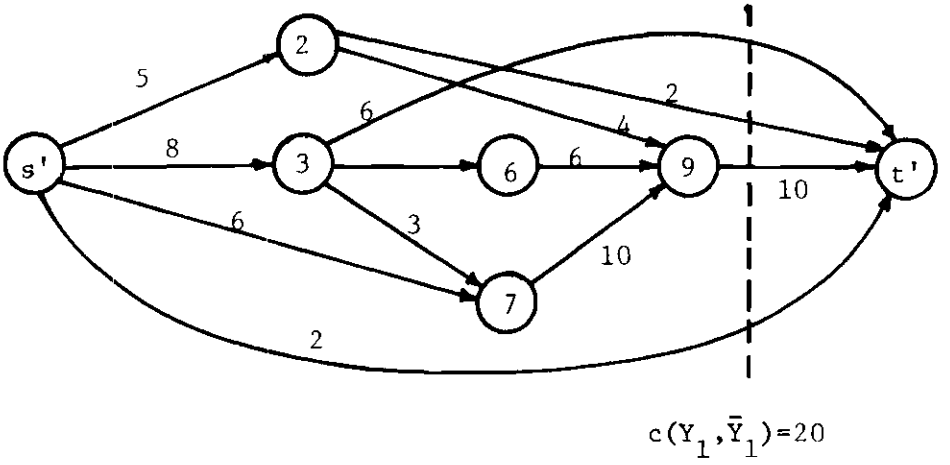
Suppose the network G is partitioned into five subnetworks as shown below:



where $(Z_i,\bar{Z}_i)$ is the min-cut for subnetwork G_i, i=1, . . . 5, and $(X_k,\bar{X}_k)$

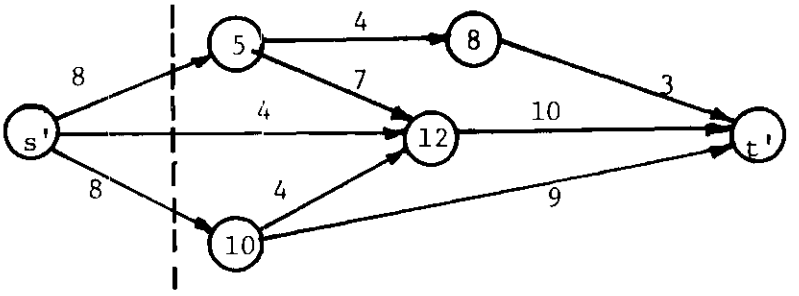
is the cut that separates subnetwork G_k from G_{k+1}, k=1, 2, . . . 4.

Step 1. Let $G = G_1 \cup G_2$. Let $(X_1,\bar{X}_1)$ divide this network into two subnetworks G_1 and G_2, respectively. We have at hand $(Z_1,\bar{Z}_1)$ and $(Z_2,\bar{Z}_2)$, the min-cuts corresponding to the subnetworks G_1 and G_2, respectively. Since $c(Z_2,\bar{Z}_2) = 20 < c(Z_1,\bar{Z}_1) = 21 < c(X_1,\bar{X}_1) = 27$, from lemma 7, the min-cut for $G_1 \cup G_2$ is either $(Z_2,\bar{Z}_2)$ or it lies between $(Z_1,\bar{Z}_1)$ and $(Z_2,\bar{Z}_2)$ and crosses $(X_1,\bar{X}_1)$. We can construct the network between $(Z_1,\bar{Z}_1)$ and $(Z_2,\bar{Z}_2)$ as shown below.



Since all the nodes are on the left of $(Y_1,\bar{Y}_1)$ this corresponds to $(Z_2,\bar{Z}_2)$, i.e., the min-cut for $G_1 \cup G_2$ is $(Y_1,\bar{Y}_1) = (Z_2,\bar{Z}_2)$.

Step 2. Consider $G_1 \cup G_2 \cup G_3$. Let $(X_2,\bar{X}_2)$ divide this network into two subnetworks $G_1 \cup G_2$ and G_3, respectively. We have the min-cut $(Y_1,\bar{Y}_1) = (Z_2,\bar{Z}_2)$ for $G_1 \cup G_2$. Since $c(Y_1,\bar{Y}_1) = c(Z_2,\bar{Z}_2) = 20 < c(Z_3,\bar{Z}_3)$ $= 22 < c(X_2,\bar{X}_2) = 27$ from lemma 7 the min-cut for $G_1 \cup G_2 \cup G_3$ is either $(Z_2,\bar{Z}_2) = (Y_1,\bar{Y}_1)$ or it lies between $(Z_2,\bar{Z}_2)$ and $(Z_3,\bar{Z}_3)$ and crosses $(X_2,\bar{X}_2)$. The network between $(Z_2,\bar{Z}_2)$ and $(Z_3,\bar{Z}_3)$ is given below.



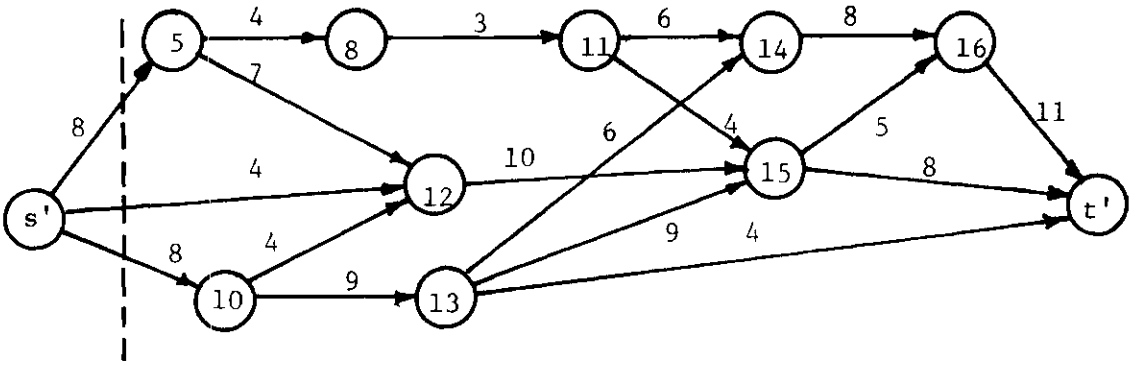
$c(Y_2,\bar{Y}_2)=20$

Since $Y_2 = \{s'\}$, the min-cut $(Y_2,\bar{Y}_2)$ does not cross $(X_2,\bar{X}_2)$ and hence $(Z_2,\bar{Z}_2)$ is the min-cut for $G_1 \cup G_2 \cup G_3$, i.e., $(Y_2,\bar{Y}_2) = (Z_2,\bar{Z}_2)$.

Step 3. Consider $G_1 \cup G_2 \cup G_3 \cup G_4$. Let $(X_3,\bar{X}_3)$ divide this network into two subnetworks $G_1 \cup G_2 \cup G_3$ and G_4, respectively. Form step 2, $(Z_2,\bar{Z}_2)$ is the min-cut for $G_1 \cup G_2 \cup G_3$, and we have at hand, the min-cut $(Z_4,\bar{Z}_4)$ for the subnetwork G_4. Since

$$c(Y_2,\bar{Y}_2) = c(Z_2,\bar{Z}_2) = 20 < c(Z_4,\bar{Z}_4) = 23 < c(X_3,\bar{X}_3) = 25$$

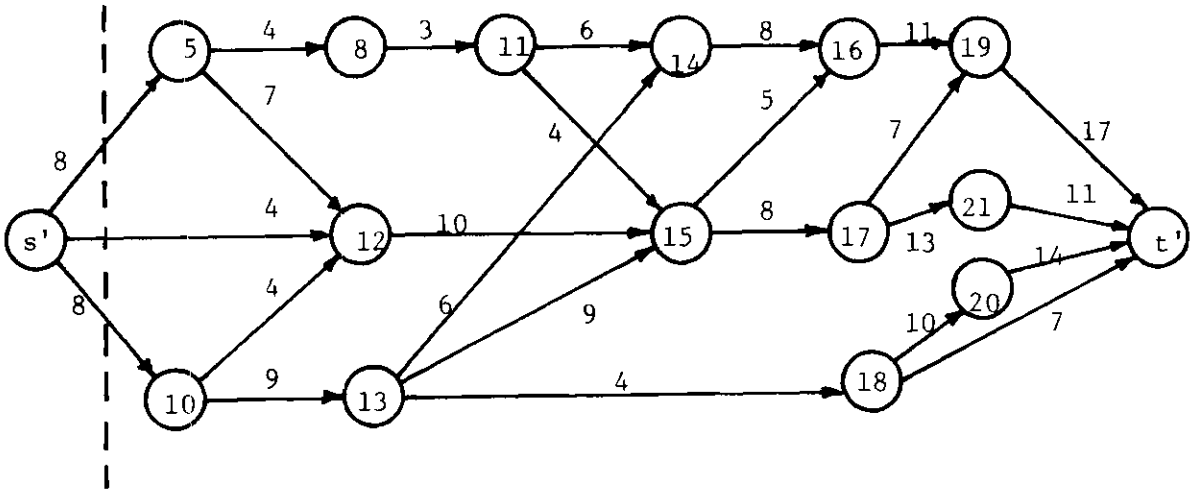
either $(Z_2,\bar{Z}_2)$ is the min-cut for $G_1 \cup G_2 \cup G_3 \cup G_4$, or the min-cut lies between $(Z_2,\bar{Z}_2)$ and $(Z_4,\bar{Z}_4)$ and crosses $(X_3,\bar{X}_3)$. Construct the network between $(Z_2,\bar{Z}_2)$ and $(Z_4,\bar{Z}_4)$.



$c(Y_3,\bar{Y}_3)=20$

The cut $(Y_3,\bar{Y}_3) = (Z_2,\bar{Z}_2)$ is the min-cut for the network $G_1 \cup G_2 \cup G_3 \cup G_4$.

Step 4. Let $G = G_1 \cup G_2 \cup G_3 \cup G_4 \cup G_5$. Let the cut $(X_4,\bar{X}_4)$ divide this network into two subnetworks $(G_1 \cup .. \cup G_4)$, G_5, respectively. The corresponding min-cuts are $(Y_3,\bar{Y}_3) = (Z_2,\bar{Z}_2)$ and $(Z_5,\bar{Z}_5)$, respectively. Since $c(Z_2,\bar{Z}_2) = 20 < c(Z_5,\bar{Z}_5) = 49 < c(X_4,\bar{X}_4) = 52$, the min-cut for G is either $(Z_2,\bar{Z}_2)$ or it lies between $(Z_2,\bar{Z}_2)$ and $(Z_5,\bar{Z}_5)$ and crosses $(X_4,\bar{X}_4)$. The network between $(Z_2,\bar{Z}_2)$ and $(Z_5,\bar{Z}_5)$ is given below.



$c(Y_4,\bar{Y}_4)=20$

The min-cut $(Y_4,\bar{Y}_4)$, corresponding to the original network, does not cross $(X_4,\bar{X}_4)$ and hence, it is $(Z_2,\bar{Z}_2)$.

Therefore, at the termination of the algorithm, the min-cut for G is found as $(Z_2,\bar{Z}_2)$ with its capacity being equal to 20.

The maximum number of nodes dealt with by fanning from one end reached 14 nodes of the original network in the last step.

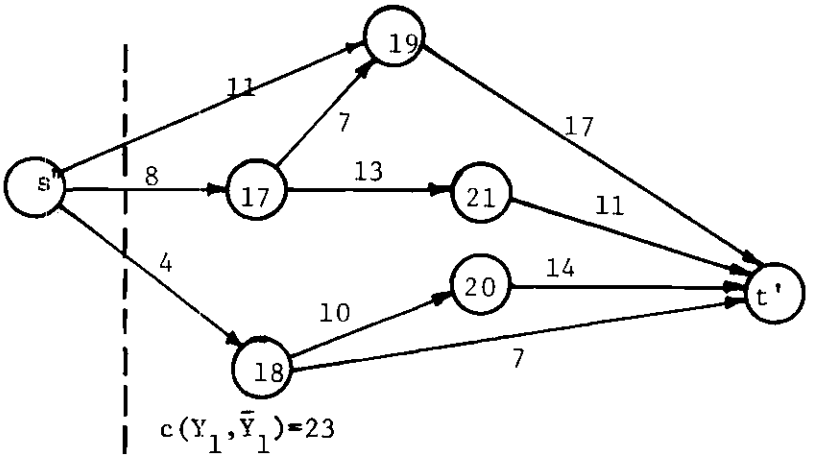
Consider solving the same problem by using blocks. We have

$$|c(Z_1,\bar{Z}_1) > c(Z_2,\bar{Z}_2)| < | c(Z_3,\bar{Z}_3) < c(Z_4,\bar{Z}_4) < c(Z_5,\bar{Z}_5)|$$

Therefore, the min-cut values for the subnetworks form 2 blocks, the first

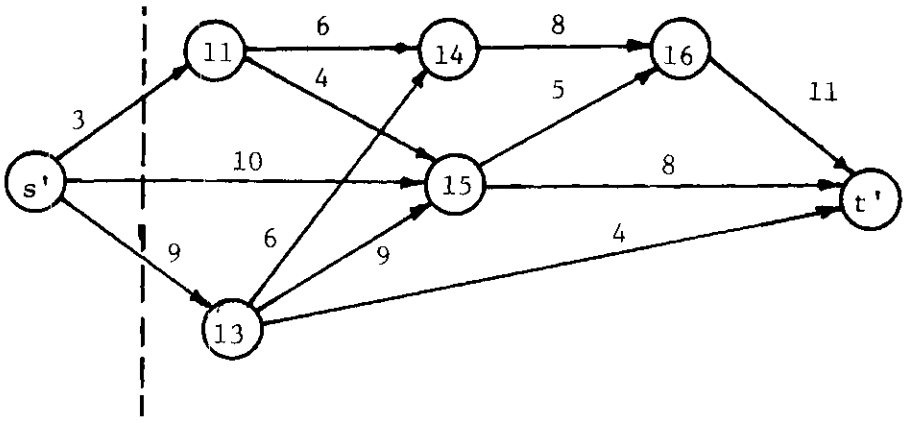
one, a decreasing and the other one, an increasing block. Choose any arbitrary block to start with, say block 2.

Step 1. Since max $\{c(Z_3,\bar{Z}_3),\ c(Z_4,\bar{Z}_4),\ c(Z_5,\bar{Z}_5)\} = c(Z_5,\bar{Z}_5)$, we start with $G_4 \cup G_5$. Let $(X_4,\bar{X}_4)$ divide this network into subnetworks G_4 and G_5, respectively. Since $c(Z_4,\bar{Z}_4) = 23 < c(Z_5,\bar{Z}_5) < c(X_4,\bar{X}_4)$, from lemma 7, either $(Z_4,\bar{Z}_4)$ is the min-cut for $G_4 \cup G_5$ or else the min-cut lies between $(Z_4,\bar{Z}_4)$, $(Z_5,\bar{Z}_5)$ and crosses $(X_4,\bar{X}_4)$. The network between $(Z_4,\bar{Z}_4)$ and $(Z_5,\bar{Z}_5)$ is given below.



Since $Y_1 = \{s'\}$, $(Z_4,\bar{Z}_4)$ is the min-cut for $G_4 \cup G_5$.

Step 2. Consider $G_3 \cup G_4 \cup G_5$. Let $(X_3,\bar{X}_3)$ divide the network into G_3 and $G_4 \cup G_5$. We have their corresponding min-cuts $(Z_3,\bar{Z}_3)$ and $(Y_1,\bar{Y}_1) = (Z_4,\bar{Z}_4)$, respectively. Since $c(Z_3,\bar{Z}_3) = 22 < c(Z_4,\bar{Z}_4) = 23 < c(X_3,\bar{X}_3) = 25$, the min-cut for $G_3 \cup G_4 \cup G_5$ is either $(Z_3,\bar{Z}_3)$ or it lies between $(Z_3,\bar{Z}_3)$ and $(Z_4,\bar{Z}_4)$ and crosses $(X_3,\bar{X}_3)$. Following is the network between $(Z_3,\bar{Z}_3)$ and $(Z_4,\bar{Z}_4)$.



$c(Y_2,\bar{Y}_2)=22$

Since $Y_2 = \{s'\}$, the min-cut for $G_3 \cup G_4 \cup G_5$ is $(Y_2,\bar{Y}_2) = (Z_3,\bar{Z}_3)$ with the capacity 22. At this point, we found the min-cut for the second block. Therefore $(Z_3,\bar{Z}_3)$ is the min-cut for the second block.

Step 3. At this step we will find the min-cut for the first block. Since we have 2 subnetworks in the block we will find the min-cut for this block in one step. Consider $G_1 \cup G_2$. Let $(X_1,\bar{X}_1)$ divide this network into two. This step is exactly the same as step 1 of the previous approach and from the previous approach we discovered that $(Y_3,\bar{Y}_3) = (Z_2,\bar{Z}_2)$ is the min-cut for $G_1 \cup G_2$.

At this point we have at hand the min-cut $(Z_2,\bar{Z}_2)$ for the first block and the min-cut $(Z_3,\bar{Z}_3)$ for the second block. Since $c(Z_2,\bar{Z}_2) < c(Z_3,\bar{Z}_3)$, they form an increasing sequence of a single block. Therefore, in the next step we will consider the network G divided into two by $(X_2,\bar{X}_2)$ in which on the left lies the first block and on the right lies the second block.

Step 4. Consider $G = \{G_1 \cup G_2\} \cup \{G_3 \cup G_4 \cup G_4\}$ and let $(X_2,\bar{X}_2)$ divide G into $G_1 \cup G_2$ and $G_3 \cup G_4 \cup G_5$, respectively. From steps 2 and

3 we have their min-cuts $(Z_2,\bar{Z}_2)$ and $(Z_3,\bar{Z}_3)$, respectively. From lemma 7, since $c(Z_2,\bar{Z}_2) = 20 < c(Z_3,\bar{Z}_3) = 22 < c(X_2,\bar{X}_2) = 27$, either $(Z_2,\bar{Z}_2)$ is the min-cut for G or else the min-cut lies between $(Z_2,\bar{Z}_2)$ and $(Z_3,\bar{Z}_3)$ and crosses $(X_2,\bar{X}_2)$. The network between $(Z_2,\bar{Z}_2)$ and $(Z_3,\bar{Z}_3)$ is given in step 2 of the previous approach. From the result of step 2 of the previous approach, we found that $(Z_2,\bar{Z}_2)$ was the min-cut. Therefore, the true min-cut for $G = G_1 \cup G_2 \cup G_3 \cup G_4 \cup G_5$ is $(Z_2,\bar{Z}_2)$ with capacity equal to 20.

The maximum number of nodes being dealt with never exceeded five in this approach.

As a conclusion, utilizing the blocks will always keep the number of nodes dealt with to a minimum.

Theoretical Efficiency

We will try to prove the theoretical efficiency of the proposed algorithms in two parts. The first will contain the proof for the best case. That is, at each step lemma 7 -iv fails to hold. The second part will give the theoretical upper bound on the number of flow augmentations in the worst case. That is, lemma 7 -iv holds.

The following theorem will be used to evaluate the efficiency of the algorithm.

Theorem 4 (11). If, in the labeling method for finding the maximum flow in a network on n nodes, each flow augmentation is performed along an augmenting path having fewest arcs, then a maximum flow (min-cut) will be obtained after no more than $\frac{1}{4}(n^3 - n)$ flow augmentations.

Case A. Lemma 7 -(iv) fails to hold at each iteration.

This case implies that lemma 7 (i), (ii) or (iii) holds at each iteration. From lemma 7, if this is true, then we do not need to do any more flow computations except the ones done for each subnetwork.

Suppose the original network G = (N,A) contains mn nodes. Suppose further, that the network is decomposed into m nonoverlapping node sets $N_1, N_2, \ldots, N_m$ by the arbitrary cuts $(X_1,\bar{X}_1), \ldots, (X_{m-1},\bar{X}_{m-1})$.

Assume that $|N_1| = |N_2| = \ldots = |N_m| = n$.

Without decomposition, the network G = (N,A) will have a theoretical upper bound of $\frac{1}{4}((mn)^3 - (mn))$ flow augmentations.

With decomposition, the upper bounds of flow augmentation for each subnetwork can be written as $\frac{1}{4}((n+1)^3 - (n+1))$ for the first and the last subnetworks and $\frac{1}{4}\{(n+2)^3 - (n+2)\}$ for the remaining subnetworks. Therefore, the total number of flow augmentations needed to locate the min-cuts for the subnetworks $G_1, \ldots G_m$ will be $\frac{1}{4}\{(m-2)[(n+2)^3 - (n+2)] + 2[(n+1)^3 - (n+1)]\}$. Since we are in case A where lemma 7 -(iv) fails to hold, we will not need any further flow computations. Therefore, the number obtained above is the theoretical upper bound for case A. Under case A the efficiency of the decomposition over no decomposition is given by the ratio

$$\lim_{n\to\infty} \frac{\frac{1}{4}(m^3n^3 - mn)}{\frac{1}{4}\{(m-2)[(n+2)^3 - (n+2)] + 2[(n+1)^3 - (n+1)]\}} \simeq m^2$$

which is a quadratic function of the number of subnetworks.

In the no decomposition case, the flow augmentation is performed over a network with mn nodes while in the subnetworks it is performed

on the network with (n+2) nodes in it. Therefore, the labeling in the overall network will take a longer time for "breakthrough" compared to the time required to reach a "breakthrough" in a subnetwork.

Case B. At each iteration, lemma 7 - (iv) holds.

We should immediately point out that in this case, there is a small probability that at the $(m-1)^{st}$ step we might be dealing with the overall network. Suppose $(Y,\bar{Y})$ is the actual min-cut in a given network G such that, $Y = \{s\}$ and $\bar{Y} = N - \{s\}$. Also assume that $c(Z_i,\bar{Z}_i) < c(X_j,\bar{X}_j)$ $\forall$ i,j. Obviously, $(Z_1,\bar{Z}_1) = (Y,\bar{Y})$. If, lemma 7 -(iv) holds at each iteration then we will need the flow augmentations on the following networks

1. $G_1, G_2, \ldots, G_m$
2. $G_1 \cup G_2, G_1 \cup G_2 \cup G_3, \ldots, G_1 \cup G_2 \cup \ldots \cup G_m$

to find out that $(Y,\bar{Y}) = (Z_1,\bar{Z}_1)$ is the min-cut for G. Note that the very last network to be dealt with is the original network. Therefore, in case B if the probability of having this situation is large, then we ought to be skeptical about the algorithm. Fortunately, this probability is very small and approaches zero as the number of nodes in each subnetwork becomes larger.

The following provides proof of the fact that the proposed algorithm is theoretically better in any case with probability greater than .999. Consider the case where G = (N,A) contains 2n nodes. Suppose the cut $(X,\bar{X})$ divides the network into two node sets N_1 and N_2 such that $|N_1| = |N_2| = n$. After performing the max-flow computations on both subnetworks G_1 and G_2, we will obtain the corresponding min-cuts $(Z_1,\bar{Z}_1)$ and $(Z_2,\bar{Z}_2)$, respectively. Suppose further, that $c(Z_1,\bar{Z}_1)$,

$c(Z_2,\bar{Z}_2) < c(X,\bar{X})$. Therefore, we are at lemma 7 - (iv) which says that either min $\{c(Z_1,\bar{Z}_1), c(Z_2,\bar{Z}_2)\}$ is the actual min-cut for G or else it lies between $(Z_1,\bar{Z}_1)$ and $(Z_2,\bar{Z}_2)$ and crosses $(X,\bar{X})$. Therefore, we need to construct the network between $(Z_1,\bar{Z}_1)$ and $(Z_2,\bar{Z}_2)$ and solve the max-flow problem for this subnetwork one more time. The result of this max-flow computation will give us the location of $(Y,\bar{Y})$, the real min-cut for $G = (N,A)$.

Let x be the number of nodes that lie between $(Z_1,\bar{Z}_1)$ and $(Z_2,\bar{Z}_2)$. Therefore, we will obtain the min-cut $(Y,\bar{Y})$ after no more than $\frac{1}{4}\{(x+2)^3 - (x+2)\}$ flow augmentations. We have already had the upper bound of $2(\frac{1}{4})\{(n+1)^3 - (n+1)\}$ flow augmentations for locating $(Z_1,\bar{Z}_1)$ and $(Z_2,\bar{Z}_2)$. Hence, the total number of flow augmentations in case B has the upper bound:

$$\tfrac{1}{4}\left\{[(x+2)^3 - (x+2)] + 2[(n+1)^3 - (n+1)]\right\}$$

In order for the decomposition algorithm to have a better worst case analysis than no decomposition, we must have the theoretical upper bound obtained above to be less than the theoretical upper bound with no decomposition. Therefore,

$$\tfrac{1}{4}\left\{[(x+2)^3 - (x+2)] + 2[(n+1)^3 - (n+1)]\right\} < \tfrac{1}{4}[(2n)^3 - 2n]$$

must hold for almost all values of n. If we rewrite the inequality,

$$(x+2)^3 - (x+2) < 6(n^3 - n^2 - n) \qquad (3.1)$$

must hold. The following table shows the feasible values of x for different values of n.

n	x	n	x	n	x
10	≤ 15	20	≤ 33	30	≤ 51
11	≤ 17	21	≤ 35	40	≤ 70
12	≤ 19	22	≤ 37	50	≤ 88
13	≤ 20	23	≤ 39	60	≤ 106
14	≤ 22	24	≤ 40	70	≤ 124
15	≤ 24	25	≤ 42	80	≤ 142
16	≤ 26	26	≤ 44	90	≤ 160
17	≤ 28	27	≤ 46	100	≤ 179
18	≤ 30	28	≤ 48	500	≤ 900
19	≤ 31	29	≤ 50	1000	≤ 1800

The entries of the table under x column for a given n value indicates the maximum value of x which will not violate the inequality 3.1. For example, for n = 10, the proposed algorithm will be superior to a no decomposition algorithm if x is less than or equal 15. Therefore, for this case, as long as $|Z_1| + |\tilde{Z}_2| \geq 5$, the proposed algorithm will be superior to a no decomposition algorithm. In other words, if $|Z_1| + |\tilde{Z}_2| \leq 4$, then the proposed algorithm will be inferior to a no decomposition algorithm (only in the upper bound for flow augmentations). This last inequality may hold for the following combinations of $|Z_1|$ and $|\tilde{Z}_2|$:

$\lvert Z_1 \rvert$	$\lvert \bar{Z}_2 \rvert$	$\lvert Z_1 \rvert + \lvert \bar{Z}_2 \rvert$
1	1	2
1	2	3
1	3	4
2	1	3
2	2	4
3	1	4

The total number of cuts in any network with (n+1) nodes, where n is odd, can be written as

$$Q = 1 + (n-1) + \frac{(n-1)!}{(n-3)!2!} + \frac{(n-1)!}{(n-4)!3!} + \ldots + \frac{(n-1)!}{\left(\frac{n-1}{2}\right)!\left(\frac{n-1}{2}\right)!}$$

$$+ \ldots + (n-1) + 1$$

Since both G_1 and G_2 contain (n+1) nodes, the total number of cuts in each subnetwork is equal to Q.

The two subnetworks are analogous to two dice in statistics, with each one having Q faces, each face corresponding to a distinct cut. Thus, the total possible combination of cuts in G_1 and G_2 can be written as $Q_1 = Q \cdot Q = Q^2$. Consider the case for n=10. As given before, if, $\lvert Z_1 \rvert + \lvert \bar{Z}_2 \rvert \leq 5$ then, the decomposition algorithm is superior to a no decomposition algorithm. Conversely, if $\lvert Z_1 \rvert + \lvert \bar{Z}_2 \rvert \leq 4$ then, a no decomposition algorithm will be superior to the proposed algorithm. As listed in the table before, this inequality can hold in 6 different

ways. For each node combination $|Z_1| = k_1$ in G_1 and $|\bar{Z}_2| = k_2$ in G_2 such that $k_1 + k_2 \leq 4$, we must evaluate the possible ways of obtaining $|Z_1| = k_1$ in G_1 and $|\bar{Z}_2| = k_2$ in G_2. Thus, the total number of combinations of having $|Z_1| = k_1$ and $|\bar{Z}_2| = k_2$ will be the product of their corresponding possible combinations. For example, when $|Z_1| = |\bar{Z}_2| = 1$, this implies that $Z_1 = \{s\}$ and $\bar{Z}_2 = \{t\}$ and each one can happen in only one way. Thus, the total number of ways of obtaining $|Z_1| = 1$ and $|\bar{Z}_2| = 1$ is given by $1 \times 1 = 1$. Consider the case $|Z_1| = 2$ and $|\bar{Z}_2| = 2$. In an (n+1) node network there are (n-1) ways of having $|Z_1| = 2$. Similarly, there are (n-1) ways of having $|\bar{Z}_2| = 2$ in G_2. Thus, the total number of combinations of $|Z_1| = 2$ and $|\bar{Z}_2| = 2$ is given by $(n-1)(n-1)=(n-1)^2$.

The following table gives all the possible combinations of $|Z_1|$ and $|\bar{Z}_2|$ which are unfavorable to the proposed decomposition algorithm. The table also contains the total possible combinations of $|Z_1|$ and $|\bar{Z}_2|$, namely, Q^2. The entries of the column f_i indicates the total possible ways of obtaining the given $|Z_1|$ and $|\bar{Z}_2|$ combination. The last column indicates the probability of an event defined by a combination of $|Z_1|$ and $|\bar{Z}_2|$. Thus, for each n, the probability of proposed algorithm being inferior is given by the sum of the probabilities corresponding to each $|Z_1|$ and $|\bar{Z}_2|$ combination given in the table. For example, the probability of the proposed algorithm being inferior to a no decomposition algorithm for n=10 is given by

$$P\left(|Z_1| + |\bar{Z}_2| \leq 4\right) = P\left(|Z_1| = 1 \wedge |\bar{Z}_2| = 1\right) + P\left(|Z_1| = 1 \wedge |\bar{Z}_2| = 2\right)$$

n	$\|Z_1\|$	$\|\bar{Z}_2\|$	No. of Com. $\|Z_1\|$ in G_1 A	No. of Com. $\|\bar{Z}_2\|$ in G_2 B	A*B = f_i	Q^2	P_i
10	1	1	1	1	1	$(512)^2$	$1/(512)^2$
	1	2	1	9	9	$(512)^2$	$9/(512)^2$
	1	3	1	36	36	$(512)^2$	$36/(512)^2$
	2	1	9	1	9	$(512)^2$	$9/(512)^2$
	2	2	9	9	81	$(512)^2$	$81/(512)^2$
	3	1	36	1	36	$(512)^2$	$36/(512)^2$
11	1	1	1	1	1	$(1024)^2$	$1/(1024)^2$
	1	2	1	10	10	$(1024)^2$	$10/(1024)^2$
	1	3	1	45	45	$(1024)^2$	$45/(1024)^2$
	2	1	10	1	10	$(1024)^2$	$10/(1024)^2$
	2	2	10	10	100	$(1024)^2$	$100/(1024)^2$
	3	1	45	1	45	$(1024)^2$	$45/(1024)^2$
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

Table 1. Possible Combinations of $|Z_1|$ and $|\bar{Z}_2|$ Unfavorable for Algorithm A, and Their Respective Probabilities

$$+ P\left(|Z_1| = 1 \wedge |\bar{Z}_2| = 3\right) + P\left(|Z_1| = 2 \wedge |\bar{Z}_2| = 1\right)$$

$$+ P\left(|Z_1| = 2 \wedge |\bar{Z}_2| = 2\right) + P\left(|Z_1| = 3 \wedge |\bar{Z}_2| = 1\right).$$

Thus,

$$P\left(|Z_1| + |\bar{Z}_2| \leq 4\right) = \frac{1 + 9 + 36 + 9 + 81 + 36}{(512)^2} = \frac{172}{(512)^2} = 0.0007$$

For n=11, the probability of inferiority can be written as

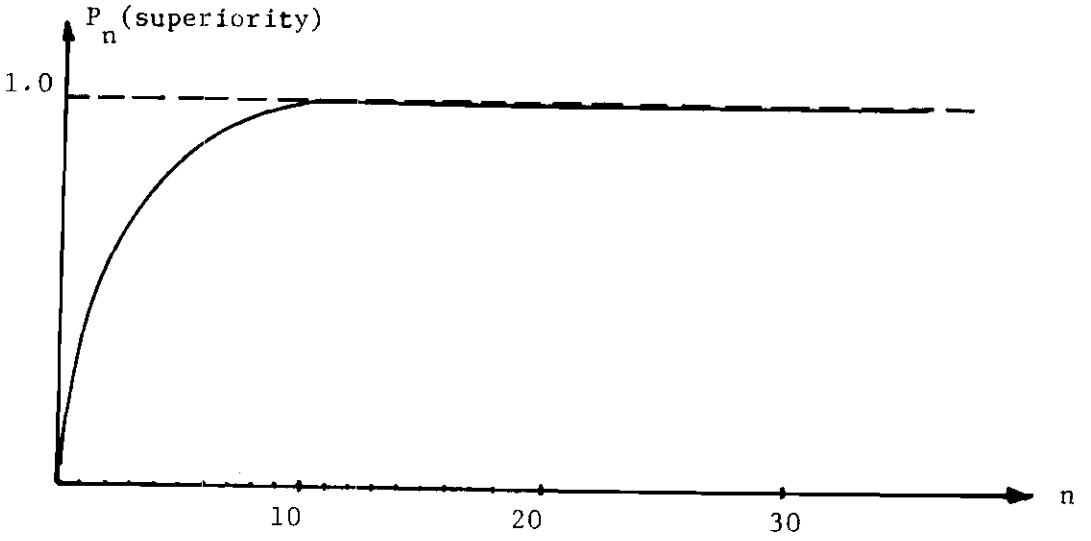
$$P\left(|Z_1| + |\bar{Z}_2| \leq 4\right) = \frac{1 + 10 + 45 + 10 + 120 + 45}{(1024)^2} = 0.00022.$$

Therefore, the probability of superiority of algorithm A for each n can be written as

$$P_n(\text{superiority}) = 1 - P_n(\text{inferiority})$$

The values of P_n(superiority) is calculated for different values of n and depicted in the following graph.

In conclusion, partioning the network into two subnetworks will be superior to a no partitioning with almost 100% certainty. Again, note that this superiority is in the upper bound of the number of flow augmentations. In addition to the flow augmentation superiority, the algorithm also has the superiority of labeling fewer nodes for each flow augmentation, since the networks considered at each iteration is a subset of the original network G = (N,A).



It is very difficult to give a mathematical upper bound in the worst case analysis on the number of flow augmentation when the network is decomposed into more than two subnetworks. We suggest that the efficiency of the algorithm to be determined by a computer simulation.

CHAPTER IV

DECOMPOSITION ON SHORTEST PATH

The basic assumption behind the decomposition of a large-scale network into m overlapping subnetworks is that of the original network being a loosely connected (sparse) network. T. C. Hu [26], Steenbrink [45] and Glover, Klingman and Napier [18] claim that such networks are abundant in real life problems.

In an n node network G = (N,A), we can have maximum n(n-1) arcs possible. If an n-node network contains n(n-1) nodes, then the network is called a 100% dense network. The density of the network can be defined as the ratio of the number of actual arcs in the network to the maximum possible number of arcs n(n-1). If this ratio is small (less than .20), the network is called sparse or loosely connected.

T. C. Hu [26] gives a method of decomposing an n-node large-scale network G = (N,A) into m linearly overlapping subnetworks. The principle of this decomposition is as follows: Let G = (N,A) be a sparse network.

Step 1. Let $N_1 \subset N$, $s \in N_1$ be a subset of nodes in G chosen arbitrarily and let i=1.

Step 2. For i=1, 2, ... , m let

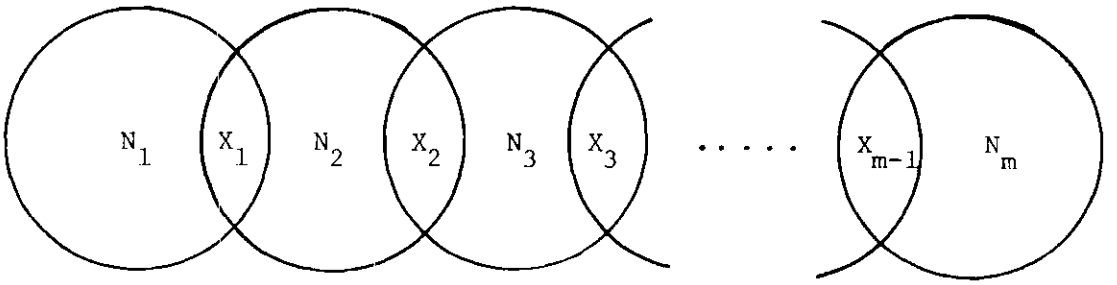
$$\bar{N}_i = \begin{cases} N_1 \cup X_1 & \text{if} \quad i=1 \\ X_{i-1} \cup N_i \cup X_i & i \neq 1,\ i \neq m \\ X_{m-1} \cup N_m & \text{if} \quad i=m \end{cases}$$

where the node-sets X_i are determined as follows:

$$X_i = \begin{cases} j/d_{kj} \neq \infty \text{ or } d_{jk} \neq \infty \text{ where } k \epsilon N_1, \ i=1 \\ j/d_{kj} \neq \infty \text{ or } d_{jk} \neq \infty \text{ where } k \epsilon N_i, \ i=2, 3, \ldots, m-1 \end{cases}$$

The node-set X_i obtained from this process is called the minimum node cut-set for N_i. Where each time an X_i is determined, N_{i+1} is chosen arbitrarily as a disconnecting set of $N_i \cup X_i$. Obviously, once X_{m-1} is determined, then the nodes $N - \{N_1 \cup X_1 \cup N_2 \cup X_2 \cup \ldots \cup N_{m-1} \cup X_{m-1}\}$ are assigned into N_m. For more detail on this decomposition see T. C. Hu [26].

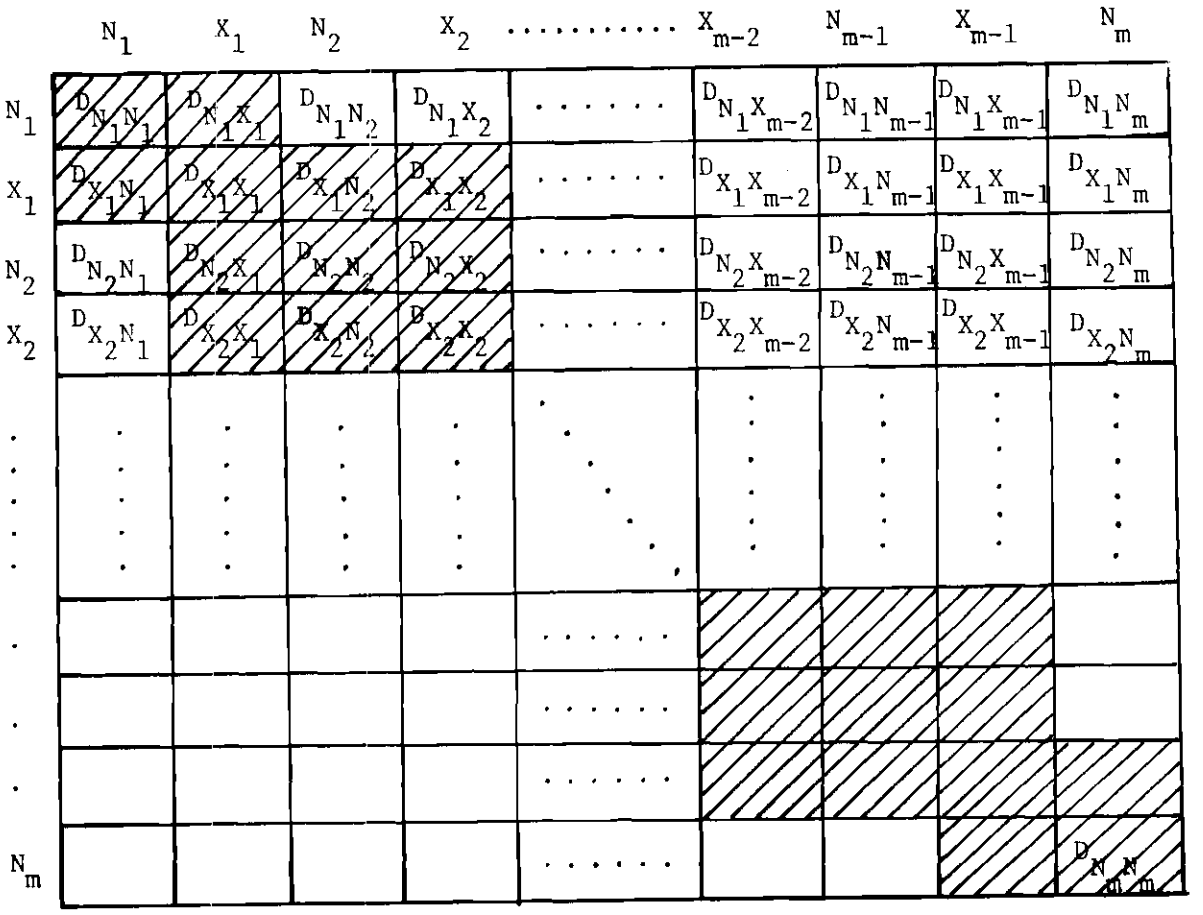
Pictorially the decomposition will look like the figure given below.



$$\text{Let } \bar{N}_1 \triangleq N_1 \cup X_1, \ \bar{N}_2 \triangleq X_1 \cup N_2 \cup X_2, \ \ldots$$

$$\bar{N}_{m-1} \triangleq X_{m-2} \cup N_{m-1} \cup X_{m-1}, \ \bar{N}_m \triangleq N_m \cup X_{m-1}.$$

The associated distance matrix can be arranged in a similar manner to depict the decomposition, as given below.



Initially only $D_{\bar{N}_i\bar{N}_i}$ i=1, ..., m have finite entries (the shaded areas). The rest of this distance matrix (unshaded areas) is set equal to the infinity.

Consider the following operation defined on the nodes of any sub-network $\bar{N}_p$.

$$d_{ik} \triangleq \min \{d_{ik}, d_{ij} + d_{jk}\} \qquad (1)$$

for all $i \neq j \neq k \epsilon \bar{N}_p$. This operation is called the <u>triple operation</u>. Finding the shortest distances between all pairs of nodes in an n-node network requires n(n-1)(n-2) applications of the triple operation (1). Since each triple operation consists of one addition and one comparison

for each i and j for a fixed k, finding all the shortest distances between all pairs of nodes in an n-node network requires $n(n-1)(n-2) \approx n^3$ additions and the same number of comparisons.

Suppose an n node network G = (N,A) is decomposed into m overlapping subnetworks as described before. Without loss of generality, suppose that $|N_1| = |N_2| = \ldots = |N_m| = u$ and $|X_1| = |X_2| = |X_{m-1}| = v$. Hence we can write $n = mu + (m-1)v$.

T. C. Hu [26] gives the following decomposition algorithm for finding the shortest paths between all pairs of nodes.

Step 1. Perform the triple operations on the subnetworks $\bar{N}_1$, $\bar{N}_2$, $\ldots$, $\bar{N}_m$ successively, where conditional shortest distances obtained in one subnetwork will replace the original distances of the succeeding network, i.e., $D_{X_iX_i}(\bar{N}_1 \cup \ldots \cup \bar{N}_i)$ will replace $D_{X_iX_i}$ before performing the triple operations on the network $\bar{N}_{i+1}$, i=1, ..., m-1. At the end of this step we will have

$$D^*_{\bar{N}_1\bar{N}_1}(\bar{N}_1),\ D^*_{\bar{N}_2\bar{N}_2}(\bar{N}_1 \cup \bar{N}_2),\ \ldots\ D^*_{\bar{N}_{m-1}\bar{N}_{m-1}}(\bar{N}_1 \cup \ldots \cup \bar{N}_{m-1}),$$

$$D^*_{\bar{N}_m\bar{N}_m}(N) \qquad \text{where } N = \bar{N}_1 \cup \bar{N}_2 \cup \ldots \cup \bar{N}_m.$$

Step 2. Perform the triple operations on the networks $\bar{N}_{m-1} \ldots \bar{N}_2$, $\bar{N}_1$ successively in reverse order of Step 1. Again the conditional shortest distances obtained in one network will replace the existing shortest distances in the preceeding network (we are performing in reverse order) before we perform the triple operations on the preceeding network. In other words $D^*_{X_kX_k}(N)$ will replace the previous distances

$D_{X_k X_k}(\bar{N}_1 \cup \bar{N}_2 \cup \ldots \cup \bar{N}_{k+1})$ obtained in Step 1 before we perform the triple operation on network $\bar{N}_k$, k = m-1, ..., 1.

At the end of this step we will have $D^*_{\bar{N}_1\bar{N}_1}(N)$, ..., $D^*_{\bar{N}_m\bar{N}_m}(N)$.

The triple operations are applied only once on the last subnetwork N_m in Step 1 of the algorithm but not in Step 2.

Step 3. Form the distance matrix

$$X = X_1 \cup X_2 \cup X_3 \cup \ldots \cup X_{m-1}$$

Perform the triple operations on this matrix X. At the end of this step we will have $D^*_{XX}(N)$, the actual shortest distances d^*_{ij} $i \epsilon X_k$, $j \epsilon X_p$, k, p = 1, 2, ..., m-1.

Step 4. Compute the shortest distances $d_{\alpha X_k}(N)$ between a node $\alpha \epsilon N_J$ and the node set X_k according to the formula

$$d^*_{\alpha X_k}(N) = \min_{\gamma \epsilon X_j} [d^*_{\alpha\gamma}(N) + d^*_{\alpha X_k}(N)]$$

where $\alpha \epsilon N_j$, $\gamma \epsilon \bar{N}_j$, $\gamma \epsilon X_j$.

Step 5. Compute the shortest distances $d^*_{ij}(N)$ $i \epsilon N_k$, $j \epsilon N_p$, $k \neq p$ according to the formula

$$d^*_{ij}(N) = \min_{X_p} [d^*_{iX_p}(N) + d^*_{X_p j}(N)]$$

The number of additions and comparisons required by the algorithm can be assigned to each step as

Step 1: $2(u+v)^3 + (m-2)(u+2v)^3$

Step 2: $(u+v)^3 + (m-2)(u+2v)^3$

Step 3: $(m-1)^3 v^3$

Step 4: $2(m-2)\ uv^2 + (m-2)(m-3)\ uv^2$

Step 5: $m(m-1)\ u^2v$

The total number of operations (additions and comparisons) is then

$$(2m-1)\ u^3 + (m^2+11m-15)\ u^2v + (m^2+12m-37)\ uv^2 + [(m-1)^3 + 16m - 29]\ v^3 \qquad (2)$$

If we apply the triple operations on the original network without the decomposition, the algorithm would require

$$n^3 = [mu + (m-1)\ v]^3 = m^3u^3 + 3m^2(m-1)\ u^2v + 3m(m-1)^2uv^2 + (m-1)^3\ v^3 \qquad (3)$$

For a large m, the ratio of (2) to (3) is given by $\left[\frac{v}{(u+v)}\right]^3$. Note, that evne though the dominating term in (2) is $(2m-1)\ u^3$, for large values of m and v being less than u but not too small, the term m^3v^3 becomes significantly important.

Realizing this fact, Hu [27] proposes a refined decomposition algorithm which is aimed at eliminating the $0(m^3v^3)$ term from the number of operations required.

The first two steps of the refined algorithm are the same as the original algorithm. Instead of Steps 3, 4, and 5 of the original algorithm the following step is proposed.

Step 3′: Find the shortest distances between any two nodes which are not both in one of the sets $\bar{N}_p$, p=1, . . ., m by the following matrix minisummation.

$$d^*_{ik} = \min_j (d^*_{ij} + d^*_{jk}) \quad j=1, \ldots v, \; i \epsilon N_{p-1}, \; k \epsilon N_p$$

where the distance matrix $d^*_{N_{p-1}X_{p-1}}(N)$ is a (u x v) matrix and the distance matrix $d^*_{X_{p-1}N_p}(N)$ is a (v x u) matrix. This whole operation can be represented by a symbol $N_{p-1} \oplus X_{p-1} \oplus N_p$. Although we need to calculate $N_p \oplus X_{p-1} \oplus N_{p-1}$ as well as $N_{p-1} \oplus X_{p-1} \oplus N_p$ we will list only one of them. Apply the matrix minisummation in the following order.

$N_1 \oplus X_1 \oplus N_2 \cup X_2$,

$N_1 \cup X_1 \cup N_2 \oplus X_2 \oplus N_3 \cup X_3$

$N_1 \cup X_1 \cup N_2 \cup X_2 \cup N_3 \oplus X_3 \oplus N_4 \cup X_4$,

.

.

.

$N_1 \cup X_1 \cup N_2 \cup \ldots \cup N_{m-2} \oplus X_{m-2} \oplus N_{m-1} \cup X_{m-1}$

$N_1 \cup X_1 \cup \ldots \cup N_{m-1} \oplus X_{m-1} \oplus N_m$.

Again, the conditional distances obtained in one matrix minisummation will be used in the second minisummation prior to the application of the triple operations in the succeeding matrix.

The number of operations needed for this modified algorithm can be listed for each step as

Step 1: $2(u+v)^3 + (m-2)(u+2v)^3$

Step 2: $(u+v)^3 + (m-2)(u+2v)^3$

Step 3: $2\{u + (2u+v) + \ldots + [(m-2)\,u + (m-3)\,v]\}\, v(u+v)$

$+ 2[(m-1)\,u + (m-2)\,v]\, vu = m(m-1)\,u^2 v$

$$+ 2(m-1)(m-2)\ uv^2 + (m-2)(m-3)\ v^3$$

Therefore, the total number of operations is

$$(2m-1)\ u^3 + (m^2 + 11m - 15)\ u^2v + (2m^2 + 18m - 35)\ uv^2 + (m^2 + 11m - 23)\ v^3 \qquad (4)$$

Compared to the original algorithm, the coefficient of v^3 has dropped from m^3 to m^2. This, of course, eliminates the worries about the case where m is large and the difference between u and v is small. The ratio of (4) to (3) for large values of m is $\left\lceil \frac{v}{u+v} \right\rceil / m$ as $m \to \infty$.

Yen [51] shows that the number of computations given by (4) can further be reduced by a clever observation. This observation modifies Step 2 of the modified T. C. Hu [27] algorithm. Note that in subnetwork $\bar{N}_k$, we have $D^*_{\bar{N}_k\bar{N}_k}(\bar{N}_1 \cup \bar{N}_2 \cup \ldots \cup \bar{N}_k)$ at the end of Step 1. In Step 2, before we perform any operation on the distance matrix, we first replace $D^*_{X_kX_k}(\bar{N}_1 \cup \ldots \cup \bar{N}_k)$ by $D^*_{X_kX_k}(N) \leq D^*_{X_kX_k}(N_1 \cup \ldots \cup N_k)$. Since the entries of $D^*_{N_kN_k}$, $D^*_{N_kX_k}$ and $D^*_{X_kN_k}$ have not been changed and $D^*_{X_kX_k}(\bar{N}_1 \cup \ldots \cup \bar{N}_k)$ has been replaced by a smaller entried matrix $D^*_{X_kX_k}(N)$, performing the triple operations on $k \epsilon N_k$ for $i, j \neq k$ will produce no smaller value for the elements in $D^*_{\bar{N}_k\bar{N}_k}$ (the same operations are done in Step 1). Hence, modify the second step of the modified T. C. Hu's algorithm to

Step 2′: Apply the triple operations on the networks $\bar{N}_{m-1}$, $\bar{N}_{m-2}$, ... $\bar{N}_1$ successively given by the formula: For each $k \epsilon X_k$ $d_{ij} = \min(d_{ik} + d_{kj}, d_{ij})$ for all $i, j \epsilon N_k \cup X_k$, i and j not both in X_k and $i, j \neq k$. This modification saves approximately $(m-1)\ u^3 + (5m-8)\ u^2v + (8m-15)\ uv^2 + (5m - 9)\ v^3$ additions and comparisons. Therefore, the total number of

additions and comparisons required by this second modification is

$$mu^3 + (m^2 + 6m - 7)\, u^2v + (2m^2 + 10m - 20)\, uv^2 + (m^2 + 6m - 14)\, v^3. \qquad (5)$$

When m and u are large, Yen's modification is twice as efficient as the modified T. C. Hu's [27] algorithm.

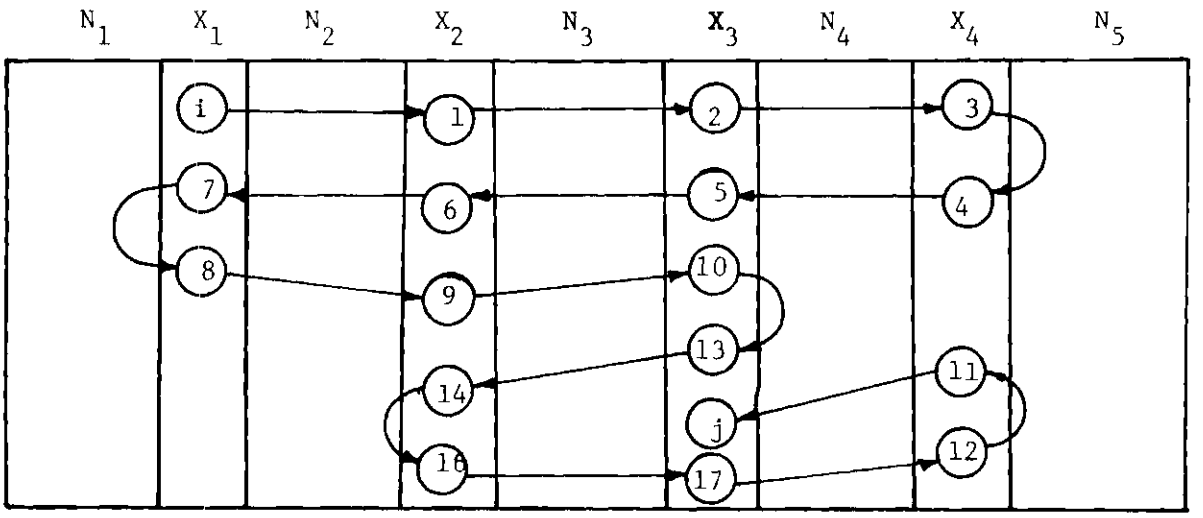
Glover, Klingman and Napier [18], suggest a different decomposition technique which reduces the lower order terms in (5) further into a smaller value. Since the network needs a very special structure, for some networks this proposed decomposition is infeasible. The details of this algorithm are given in Chapter II.

Developed Theory

Upon close examination of the construction of the network (matrix) $X = X_1 \cup X_2 \cup \ldots \cup X_{m-1}$ we realize that at the end of Step 1 of T. C. Hu's [26] original algorithm we obtain $D^*_{X_1X_1}(\bar{N}_1)$, $D^*_{X_1X_2}(\bar{N}_1 \cup \bar{N}_2)$, $D^*_{X_2X_2}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$, $\ldots$ $D^*_{X_{m-2}X_{m-1}}(\bar{N}_1 \cup \ldots \cup \bar{N}_{m-1})$, $D^*_{X_{m-1}X_{m-1}}(N)$. If we construct a matrix $X = X_1 \cup \ldots \cup X_{m-1}$ at the end of Step 1 with the entries as given above, and perform the triple operations on this matrix, the result will give us $D^*_{XX}(N)$.

Theorem 5. The shortest distances obtained in the matrix X as described above corresponds to the actual shortest distances of the original network between the nodes i and j, i, j$\in$X.

Proof. Assume that the shortest distance between the nodes i and j, i, j$\in$X is as given below.



The shortest path between $i \epsilon X_1$ and $j \epsilon X_3$ is composed of the subpath i-1 which lies completely in $\bar{N}_2$, the subpath 1-2 which lies completely in $\bar{N}_3$, ..., and the subpath 11-j which lies completely in $\bar{N}_4$.

By applying the triple operations on $\bar{N}_1, \bar{N}_2, \ldots, \bar{N}_m$ as described in Step 1 of Hu's [26] algorithm we will obtain the following distances in the given order.

$d^*_{78}(\bar{N}_1)$, $d^*_{i,1}(\bar{N}_1 \cup \bar{N}_2)$, $d^*_{6,7}(\bar{N}_1 \cup \bar{N}_2)$, $d^*_{6,8}(\bar{N}_1 \cup \bar{N}_2)$,

$d^*_{8,9}(\bar{N}_1 \cup \bar{N}_2)$, $d^*_{7,9}(\bar{N}_1 \cup \bar{N}_2)$, $d^*_{6,9}(\bar{N}_1 \cup \bar{N}_2)$, $d^*_{14,16}(\bar{N}_1 \cup \bar{N}_2)$

$d^*_{12}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$, $d^*_{56}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$, $d^*_{59}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$,

$d^*_{5,10}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$, $d^*_{6,10}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$, $d^*_{13,14}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$

$d^*_{13,16}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$, $d^*_{13,17}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$, $d^*_{14,17}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$,

$d^*_{2,3}(N\text{-}N_5)$, $d^*_{45}(N\text{-}N_5)$, $d^*_{4,10}(N\text{-}N_5)$, $d^*_{4,13}(N\text{-}N_5)$, $d^*_{11,j}(N\text{-}N_5)$

$d^*_{4,17}(N\text{-}N_5)$, $d^*_{4,12}(N\text{-}N_5)$, $d^*_{34}(N)$, $d^*_{3,12}(N)$,

$d^*_{3,11}(N)$, $d^*_{12,11}(N)$.

Therefore, we have already at hand $d^*_{i,1}(N)$, $d^*_{12}(N)$, $d^*_{2,3}(N)$, $d^*_{3,11}(N)$, $d^*_{11,j}(N)$ as well as other information. Hence,

$$d^*_{ij}(N) = d^*_{i1}(N) + d^*_{12}(N) + d^*_{23}(N) + d^*_{11,j}(N) + d^*_{3,11}(N)$$

where i, 1, 2, 3, jϵX, and $d^*_{i1}(N) = d^*_{i1}(\bar{N}_1 \cup \bar{N}_2)$, $d^*_{12}(N) = d^*_{12}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$, $d^*_{23}(N) = d^*_{23}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3 \cup \bar{N}_4)$ etc., are already obtained in Step 1 and are contained in the matrix D^*_{XX}. QED

One of the most important problems in network flows is to find the shortest path between the source s and the sink t in a given single commodity network. Most of the algorithms which give more information than the shortest path between s and t are not intentionally designed for the extra information, but rather the extra information is a natural outcome of the technique applied. For example, if we are only interested in the shortest path between s and t, none of the algorithms can supply us with that specific information but nothing else. Tree building algorithms provide the shortest paths from source s to all other nodes (whether needed or not) and matrix algorithms provide all the shortest distances. We might say that we are paying the cost of this extra information by applying the known algorithms.

Suppose we concentrate on finding the shortest path between the source s and the sink t by decomposition, but nothing else. This question leads us into the analysis of the matrix X.

Consider a new matrix X′ such that $X' = s \cup X_1 \cup X_2 \cup \ldots X_{m-1} \cup t$. That is, the matrix X as defined before with the columns and rows corresponding to node s and node t is added to it. Where the entries

of the matrix X' are defined before for the nodes $i\epsilon X_p$, $j\epsilon X_k$ p, k=1, 2, ..., m-1. The entries for the arcs (s, X_1) and (X_1,s) as well as (X_{m-1},t) and (t,X_{m-1}) are obtained from Step 1 of Hu's [26] algorithms as $D^*_{sX_1}(\bar{N}_1)$, $D^*_{X_1 s}(\bar{N}_1)$, $D^*_{X_{m-1},t}(N)$ and $D^*_{t,X_{m-1}}(N)$, respectively.

Theorem 6: The shortest distance between s and t for the original network can be obtained by utilizing X' matrix as described above.

Proof: The shortest path from s to t has to start at node s and follow the subpath s-1, $i\epsilon X_1$ and leave $\bar{N}_1$ for the first time from node $i\epsilon X_1$. No matter what the rest of the shortest path does after leaving node i, it has to enter the last subnetwork $\bar{N}_m$ the last time from a node k, and then reach the sink t via the subpath k-t which lies in $\bar{N}_m$. From Theorem 5 we know that the shortest distances between $i\epsilon X_p$ and $j\epsilon X_k$ p, k=1, ... , m-1 can be obtained from the X matrix. X is a subset of X' and hence we can obtain the shortest distances between $i\epsilon X_p$ and $j\epsilon X_k$ p, k, 1 ... , m-1 in X' matrix. From Step 1 of T. C. Hu's algorithms we have $D^*_{sX_1}(\bar{N}_1)$ and $D^*_{X_{m-1}t}(N)$ already stored in X' matrix. Therefore

$$d^*_{st}(N) = \min_{\substack{i\epsilon X_1 \\ j\epsilon X_{m-1}}} \left\{ d^*_{si}(N_1) + d^*_{ij}(N) + d^*_{jt}(N) \right\}$$

is the required equation for determining the shortest path from X' matrix. QED.

If all the entries of X' matrix are nonnegative we can use Dijkstra's [9] algorithm to obtain the shortest distances from s to $j\epsilon X'$. If the entries are arbitrary (provided no negative cycle exists) we can either use triple operations or Yen's [52] shortest path algorithm.

If we can use Dijkstra's algorithm we will need $[(m-1)v+2]^2$ additions and comparisons. If we apply triple operations, then we will need $[(m-1)v+2]^3$ additions and comparisons. In the case that Yen's [52] algorithm is used it will take $\frac{[(m-1)\ v+2]^3}{2}$ steps to find the shortest distance between s and t.

In any case we are paying the cost for the extra information, since each of those algorithms will supply us with more information than is needed.

There is one further modification on the X' matrix needed for eliminating the extra information. The matrix X' corresponds to a network $G'' = (N',A'')$ where $N' = s \cup X_1 \cup X_2 \cup \ldots \cup X_{m-1} \cup t$ and

$$A'' = \left\{(s,X_1) \cup (X_1,X_1) \cup (X_1,X_2) \cup (X_2,X_1) \cup \ldots \cup (X_{m-1},t)\right\}$$

and the costs of those arcs are the entries of X' matrix as described before.

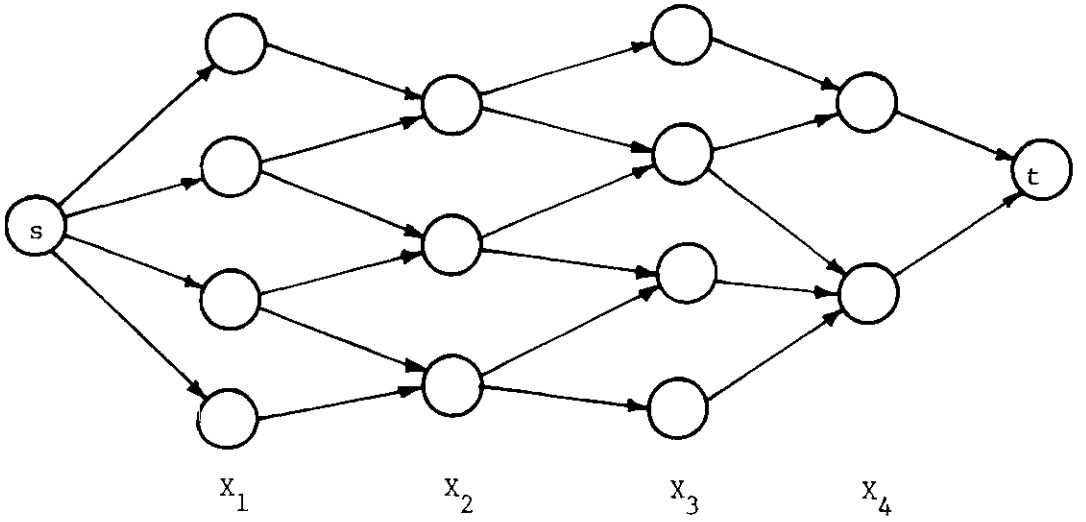
Upon considering the $G'' = (N',A'')$ network it is understood that the arcs in the form (X_i,X_i) i=1, ..., m-1 and (X_{i+1},X_i) i = m-2, ... , 1 are only helping to find the shortest distances between the nodes s and $j \neq t$ in Dijkstra's or Yen's algorithm. In the case of triple operations being applied on $G'' = (N',A'')$ network, those arcs were only helping to find $d^*_{ij}(N)$ for all i and all $j \neq t$.

The conclusion was to drop the arcs from the network G'' which are in the form (X_i,X_i) or (X_{i+1},X_i). This conclusion is crystallized in the following theorem.

Let $G' = (N',A')$ be a network obtained from $G'' = (N',A'')$ such that G' and G'' contain the same nodes, and A' is obtained from A'' by deleting the arcs in the form (X_i,X_i) i=1, ... , m-1 or (X_{i+1},X_i) i= m-2, ... , 1. The distance matrix corresponding to $G' = (N',A')$ is exactly the same as the distance matrix for G'' except all the entries for (X_i,X_i) and (X_{i+1},X_i) are set equal to infinity.

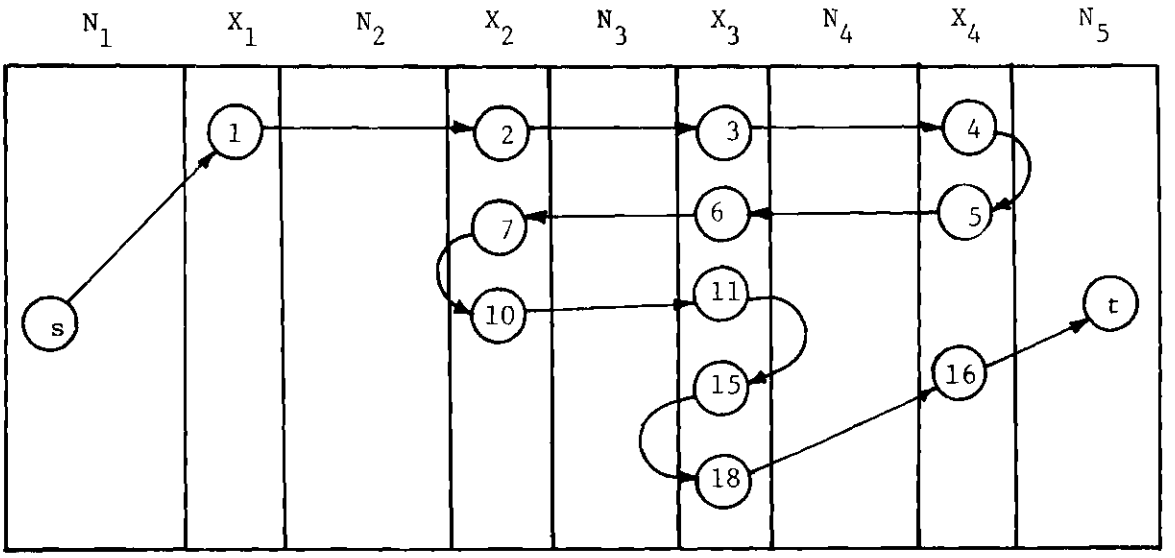
Property: $G'=(N',A')$ is an acyclic network.

Obviously G' contains the arcs in the form (s,X_1), (X_1,X_2), (X_2,X_3), ..., (X_{m-1},t). Following is a typical example for G' network.



Theorem 7: The shortest distance obtained in $G' = (N',A')$ network, between nodes s and t, corresponds to the actual shortest distance for the original network $G = (N,A)$.

Proof. Let the actual shortest path in G be as shown below. From Step 1 of Hu's algorithms we will have $d^*_{s1}(\bar{N}_1)$, $d^*_{12}(\bar{N}_1 \cup \bar{N}_2)$, $d^*_{7,10}(\bar{N}_1 \cup \bar{N}_2)$, $d^*_{23}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$, $d^*_{67}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$, $d^*_{6,10}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$, $d^*_{6,11}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$, $d^*_{15,18}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3)$, $d^*_{3,4}(N-N_5)$, $d^*_{56}(N-N_5)$, $d^*_{5,11}(N-N_5)$, $d^*_{5,15}(N-N_5)$, $d^*_{5,18}(N-N_5)$, $d^*_{5,16}(N-N_5)$, $d^*_{4,5}(N)$, $d^*_{4,16}(N)$, $d^*_{4,t}(N)$.


N_1
X_1
N_2
X_2
N_3
X_3
N_4
X_4
N_5
s
1
2
3
4
5
6
7
10
11
15
16
18
t

Therefore, G' network will contain the arcs (s,1), (1,2) (2,3), (3,4) and (4,t) as well as others. The distances on those arcs are the values obtained in Step 1 of Hu's [26] algorithm. Since the subpath s-1 lies completely in $\bar{N}_1$, then $d^*_{s1}(\bar{N}_1) = d^*_{s1}(N)$. Similarly, $d^*_{12}(\bar{N}_1 \cup \bar{N}_2) = d^*_{12}(N)$, $d^*_{23}(\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_3) = d^*_{23}(N)$, $d^*_{34}(N-N_5) = d^*_{34}(N)$. We also have $d^*_{4t}(N)$. Therefore, the shortest path is

$$d^*_{st}(N) = d^*_{s1}(N) + d^*_{12}(N) + d^*_{23}(N) + d^*_{34}(N) + d^*_{4t}(N)$$

which are all contained in ghe G' network. Thus, the shortest distance obtained in the G' network correspond to the actual shortest distance in the original network G = (N,A). QED.

The special structure of the G' network and its acyclic property makes it easier to obtain the shortest distance from s to t in a more efficient manner. The arcs of the network are in the form (s,X_1), (X_i, X_{i+1}) or (X_{m-1},t). Irrespective of the sign of the arc distances in the G' network, the following algorithm will provide the shortest distance and the path on G' network.

Algorithm B.

Initialization Step: Set $w'_s = 0$, $w'_i = d^*_{si}$, $i \epsilon X_1$, $k = 1$.

Main Step: (1). Let $k = k+1$. If $k = m$ go to the final step. Otherwise, let $X = X_k$. Go to Step 2.

(2). Let $w'_p + d^*_{pq} = \min_{i \epsilon X_{k-1}} \{w'_i + d^*_{iq}\}$ for each $q \epsilon X$. Set $w'_q = w'_p + d^*_{pq}$, and let $X = X-q$. If $X \neq \emptyset$ go to Step 2, otherwise go to 1.

Final Step: Let $w'_p + d^*_{pt} = \min_{i \epsilon X_{m-1}} \{w'_i + d^*_{it}\}$

Set $w'_t = w'_p + d^*_{pt}$. Stop. The shortest path from s to t is at hand.

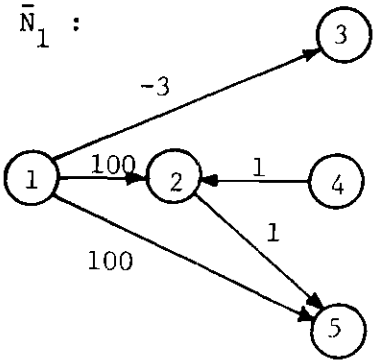
Each time the main step is executed, the algorithm will require v^2 additions and v^2 comparisons. Since the main step is executed exactly m-2 times, it will require $(m-2)v^2$ additions and $(m-2)v^2$ comparisons to complete (m-2) of the main step. The final step will need only v additions and v comparisons. Therefore, this algorithm will require $(m-2)\ v^2 + v$ additions and comparisons. The same distance is obtained on G'' network with $[(m-1)\ v+2]^3$ additions and comparisons by using Floyd's algorithm [15] for arbitrary arc weights, and $[(m-1)\ v+2]^2$ additions and comparisons by using Dijkstra's algorithm [9] in the case that the arc weights are nonnegative. Therefore, the proposed algorithm is m times more efficient than Dijkstra's algorithm in the case that the arc weights on G' are nonnegative and m^2 times more efficient than Floyd's algorithm in the case of arbitrary arc weights. This efficiency will be clearer in the next chapter when max-flow min-cost problems are handled by flow augmentation.

At this point we are ready to give the statement of the shortest path algorithm by decomposition. Suppose the given network G = (N,A) has been decomposed into m overlapping networks $\bar{N}_1, \bar{N}_2, \ldots, \bar{N}_m$, as described before.

Algorithm C.

Step 1. Perform the triple operations on subnetworks $\bar{N}_1, \bar{N}_2, \ldots, N_m$ successively, as described in Hu's [26] algorithm, where the conditional shortest distances obtained in one network will replace the original distances of the succeeding network before the triple operations

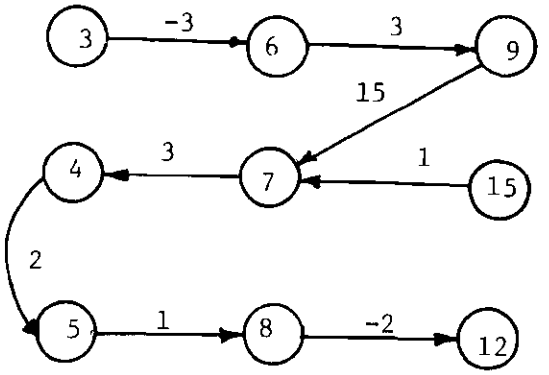
$\bar{N}_1$:



$$D^{*}_{\bar{N}_1\bar{N}_1}(\bar{N}_1) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 100 & -3 & +\infty & 100 \\ +\infty & 0 & +\infty & +\infty & 1 \\ +\infty & +\infty & 0 & +\infty & +\infty \\ +\infty & 1 & +\infty & 0 & 2 \\ +\infty & +\infty & +\infty & +\infty & 0 \end{bmatrix} \end{array}$$

Before operating on $\bar{N}_2$ we will replace $D_{X_1X_1}$ with $D_{X_1X_1}(\tilde{N}_1)$.

$\bar{N}_2$:



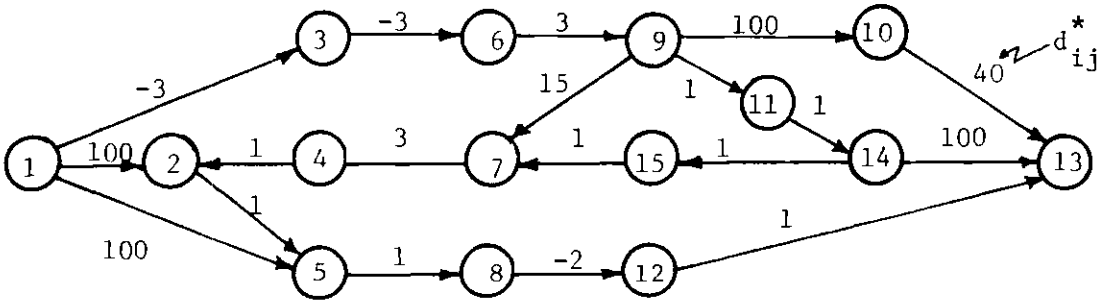
are performed on the succeeding network.

Step 2. Construct the $G' = (N', A')$ network as described before. On the nodes of G' network apply algorithm B to find the shortest distance between s and t which corresponds to the actual shortest distance on the original network G.

The following is an example of the application of algorithm C on an arbitrary cost network, for finding the shortest path between source s, and sink t.

Example on the Shortest Path Algorithm

Consider the following network.



Let $N_1 = \{1,2\}$, $X_1 = \{3,4,5\}$, $N_2 = \{6,7,8\}$, $X_2 = \{9,15,12\}$ and $N_3 = \{10,11,14,13\}$.

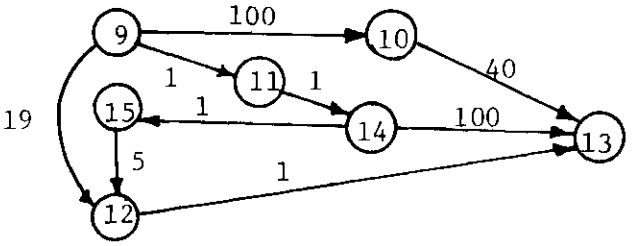
For $\bar{N}_1 = N_1 \cup X_1$ as depicted below, the conditional shortest distances are shown in the matrix, $D^*_{\bar{N}_1 \bar{N}_1}(\bar{N}_1)$.

$$D^{*}_{\bar{N}_2\bar{N}_2}(\bar{N}_1 \cup \bar{N}_2) =$$

	3	4	5	6	7	8	9	12	15
3	0	18	20	-3	15	21	0	19	$+\infty$
4	$+\infty$	0	2	$+\infty$	$+\infty$	3	$+\infty$	1	$+\infty$
5	$+\infty$	$+\infty$	0	$+\infty$	$+\infty$	1	$+\infty$	-1	$+\infty$
6	$+\infty$	21	23	0	18	24	3	22	$+\infty$
7	$+\infty$	3	5	$+\infty$	0	6	$+\infty$	4	$+\infty$
8	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	0	$+\infty$	-2	$+\infty$
9	$+\infty$	18	20	$+\infty$	15	21	0	19	$+\infty$
12	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	0	$+\infty$
15	$+\infty$	4	6	$+\infty$	1	7	$+\infty$	5	0

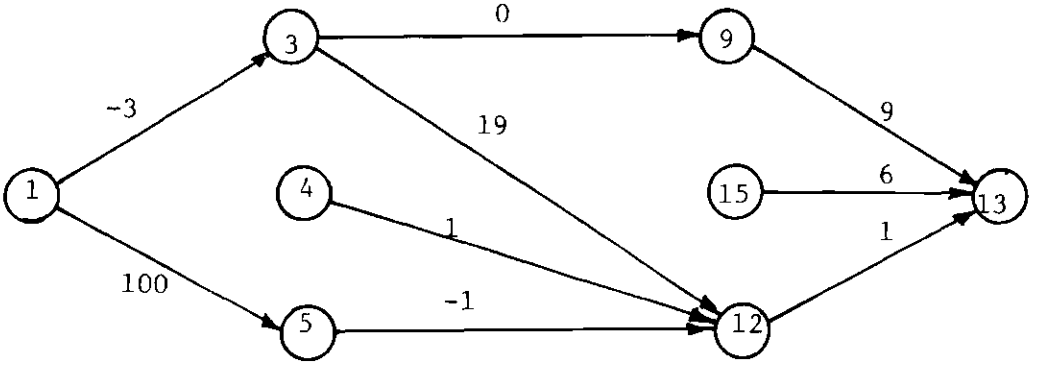
Upon replacing the conditional distances in $\bar{N}_3$ we have

$\bar{N}_3$:



$$D^*_{\bar{N}_3\bar{N}_3}(N) = \begin{array}{c} \\ 9 \\ 15 \\ 12 \\ 10 \\ 11 \\ 14 \\ 13 \end{array} \begin{array}{c} \begin{array}{ccccccc} 9 & 15 & 12 & 10 & 11 & 14 & 13 \end{array} \\ \begin{bmatrix} 0 & 3 & 8 & 100 & 1 & 2 & 9 \\ +\infty & 0 & 5 & +\infty & +\infty & +\infty & 6 \\ +\infty & +\infty & 0 & +\infty & +\infty & +\infty & 1 \\ +\infty & +\infty & +\infty & 0 & +\infty & +\infty & 40 \\ +\infty & 2 & 7 & +\infty & 0 & 1 & 8 \\ +\infty & 1 & 6 & +\infty & +\infty & 0 & 7 \\ +\infty & +\infty & +\infty & +\infty & +\infty & +\infty & 0 \end{bmatrix} \end{array}$$

The G' network will be



Hence, the shortest path is 1-3-9-13 with length 6 which corresponds to the actual path 1-3-6-9-11-14-15-7-4-2-5-8-12-13.

Theoretical Efficiency

The computational upper bound on the number of additions and comparisons required by algorithm C can be easily calculated.

Step 1. $(m-2)[u+2v]^3 + 2(u+v)^3$

Step 2. $(m-2)\ v^2 + v$

Therefore, the total number of additions and comparisons required by the algorithm C can be written as

$$mu^3 + (6m-6)\ u^2v + (12m-8)\ uv^2 + (8m-14)\ v^3 + (m-2)\ v^2 + v$$

If no decomposition is used, the same solution can be achieved by no more than $1/2[mu + (m-1)\ v]^3$ additions and comparisons by utilizing Yen's [52] algorithm. For $u \gg v$ the ratio of the number of operations in decomposition to no-decomposition can be approximated by

$$\frac{m^3u^3/2}{mu^3} = \frac{m^2}{2}$$

Therefore, the proposed algorithm C is quadratically more efficient than Yen's [52] algorithm.

CHAPTER V

MINIMUM COST MAXIMUM FLOW

This chapter deals with single-commodity minimum cost maximum flow problems. For these problems, a nonnegative function f_{ij} ranging over all arcs (i,j) of G = (N,A) is called a flow if

(i) for every (i,j) ϵA $\quad f_{ij} \leq u_{ij}$

(ii) for every node iϵN

$$\sum_i f_{ij} - \sum_i f_{ji} = \begin{cases} f & \text{if} \quad i=s \\ 0 & \text{if} \quad i\neq s,t \\ -f & \text{if} \quad i=t \end{cases}$$

The minimum cost maximum flow is a flow f* in G such that the net amount of flow f* from s to t is maximum with the minimum cost.

Let f be a flow in G = (N,A). Define a new network G^f = (N,A′) such that G^f contains the same nodes as G and if (i,j) ϵA and $u_{ij} > 0$, then (i,j) ϵA′ or if (j,i) ϵA and $f_{ji} > 0$, then (i,j) ϵA′. Define the cost of an arc on G^f as

$$d_{ij} = \begin{cases} c_{ij} & \text{if} \quad (i,j) \epsilon A \\ -c_{ij} & \text{if} \quad (j,i) \epsilon A \end{cases}$$

Define the upper capacities of each arc (i,j) on G^f as

$$u'_{ij} = \begin{cases} u_{ij} - f_{ij} & \text{if} \quad (i,j) \in A \\ \\ f_{ji} & \text{if} \quad (j,i) \in A \end{cases}$$

The network $G^f = (N,A')$, as defined above, is called the marginal cost network.

The flow augmenting algorithms for finding the minimum cost maximum flow start initially with zero flow f^o and increment the flow along the shortest path obtained in the marginal cost network G^f with respect to the arc costs d_{ij}. All those algorithms stop with the optimal solution when no flow augmenting path exists on the marginal cost network G^f with respect to flow f. Then the current flow f is said to be the minimum cost maximum flow for the network $G = (N,A)$.

Theorem 8. (Edmonds and Karp) Let f be a flow. Then the following are equivalent:

(i) f is extreme,

(ii) every directed cycle in G^f has nonnegative weights

(iii) there exists a labeling function π such that for every arc $(i,j) \in A'$

$$\pi_i + d_{ij} - \pi_j \geq 0. \tag{6}$$

If the flow of f and the labeling function π together satisfy (6), then f and π are said to be compatible. The following theorem gives the basic requirement for the flow augmentation algorithms.

Theorem 9. (Ford and Fulkerson) If f is extreme and P is a path

of shortest length in G^f from s to t, then a flow f′ obtained by augmenting along P is extreme.

Let f be a flow and let π be a labeling function. Assign each arc (i,j) of G^f a weight $\bar{d}_{ij} = \pi_i + d_{ij} - \pi_j$. Then

(i) If C is a directed cycle in G^f, then

$$\sum_{(i,j)\ C} \bar{d}_{ij} = \sum_{(i,j)\ C} d_{ij}$$

(ii) If P is a path from u* to v*, then

$$\sum_{(i,j)\ \epsilon P} \bar{d}_{ij} = \pi_{u^*} - \pi_{v^*} + \sum_{(i,j)\ P} d_{ij}$$

The following observation is due to Edmonds and Karp [11] which states: if f and π are compatible then $\bar{d}_{ij} \geq 0$ and P is the shortest path from s to t with respect to weights d_{ij} if, and only if, P is a shortest path with respect to nonnegative weights $\bar{d}_{ij}$. Therefore, the flow augmenting path on G^f can be found by utilizing the nonnegative arc weights $\bar{d}_{ij}$.

This, of course, will speed up the determination of the shortest path since Dijkstra's algorithm is extremely efficient for nonnegative cost shortest path problems.

Obviously, if all the flow capacities are integer, then the maximum number of flow augmentations is bounded by f*, the value of the maximum flow, for finding the minimum cost maximum flow. If all arc costs c_{ij}, are integers then Edmonds and Karp [11] show that the optimal flow will be obtained in, at most, $1 + 1/4(n^3 - n)(n-1)\ D$. Where n is the number of nodes in the network and $D = \max (c_{ij})$ for all $(i,j) \epsilon A$.

The following algorithm is proposed by Edmonds and Karp [11] for minimum cost flow problems in a network G = (N,A).

Step 1. Set f^o equal to zero flow and set π^o equal to the identically zero labeling function.

Step 2. Given f^k and π^k, determine f^{k+1} by augmenting along a shortest path from s to t in G^{fk} with respect to the nonnegative weights

$$\bar{d}^k_{ij} = \pi^k_i + d^k_{ij} - \pi^k_j$$

where d^k_{ij} is the cost of the arc (i,j) on the marginal cost network G^{fk} as defined before.

Step 3. If w^k_i denotes the weight of a shortest path from s to i with respect to weights $\bar{d}^k_{ij}$, set $\pi^{k+1}_i = \pi^k_i + w^k_i$; take $w^k_i = \pi^{k+1}_i = +\infty$ if i is inaccessible from s in G^{fk}.

Step 4. Stop when, for some k, no flow augmenting path exists with respect to f^k.

The properties of this algorithm are given in the following theorem which will be used in the development of the decomposition algorithm utilizing the similar ideas.

Theorem 10. (Edmonds and Karp) For each k, f^k and π^k are compatible. For each k and u, π^k_u gives the weight of a shortest path from s to u in G^{fk} with respect to the weights d^k_{ij} and $\pi^{k+1}_i \geq \pi^k_i$ for all i.

Developed Theory

The efficiency of the flow augmenting algorithm heavily depends on the shortest path algorithm which is used to determine the flow

augmenting paths. If two different flow augmenting algorithms, say X and Y, detect the same path for each flow augmentation, then suppose it takes kT additions and comparisons to detect the shortest path in algorithm Y and T additions and comparisons by algorithm X. The number of additions and comparisons needed by each algorithm can be written as kT* (no. of flow augmentations) for algorithm Y and T* (no. of flow augmentations) for algorithm X. Therefore, the relative efficiency of algorithm X over algorithm Y is given by

$$\frac{kT^* \text{ (no. of flow augmentations)}}{T^* \text{ (no. of flow augmentations)}} = k$$

which is the relative efficiency of their respective shortest path algorithms.

In the case of large-scale network flow problems the time saved by each flow augmentation becomes substantial. For networks over 1000 nodes and several thousand arcs, most of the shortest path algorithms become inapplicable because of the storage requirements. At this point the decomposition on shortest path problems becomes important.

Algorithm C given in Chapter IV becomes vital in large-scale minimum cost flow problems. The decomposition enables us to handle the smaller size networks at any time of the flow augmentation. At the same time the flow augmenting path is theoretically expected to be obtained much faster compared to the current shortest path algorithms.

Let G = (N,A) be decomposed into m overlapping networks $\bar{N}_1$, $\bar{N}_2$, ... , $\bar{N}_m$ as defined before. For a given f in G, define the marginal cost networks $\bar{N}_1^f$, $\bar{N}_2^f$, ... , $\bar{N}_m^f$ as: $\bar{N}_j^f$ contains the same nodes as in $\bar{N}_j$, j=1,

..., m. $(i,j) \in \bar{N}_k^f$ if $(i,j) \in \bar{N}_k$ and $u_{ij} - f_{ij} > 0$ for k=1, ..., m or $(i,j) \in \bar{N}_k^f$ if $(j,i) \in \bar{N}_k$ and $f_{ij} > 0$. Define the cost of an arc $(i,j) \in \bar{N}_k^f$, k=1, ..., m as

$$d_{ij} = \begin{cases} c_{ij} & \text{if} \quad (i,j) \in \bar{N}_k \\ -c_{ij} & \text{if} \quad (j,i) \in \bar{N}_k. \end{cases}$$

Define the upper capacities of each arc $(i,j) \in \bar{N}_k^f$, k=1, ..., m as

$$u'_{ij} = \begin{cases} u_{ij} - f_{ij} & \text{if} \quad (i,j) \in \bar{N}_k \\ f_{ji} & \text{if} \quad (j,i) \in \bar{N}_k \end{cases}$$

By using the procedure described above, we created m marginal cost networks corresponding to $\bar{N}_1, \bar{N}_2, \ldots, \bar{N}_m$ with respect to the given flow f.

The following is the minimum cost maximum flow algorithm by decomposition.

Algorithm D

Let G = (N,A) be decomposed into m overlapping subnetworks $\bar{N}_1, \ldots, \bar{N}_m$ as described before. Let $f^o = 0$ and $d_{ij} = c_{ij}$ for $(i,j) \in A$.

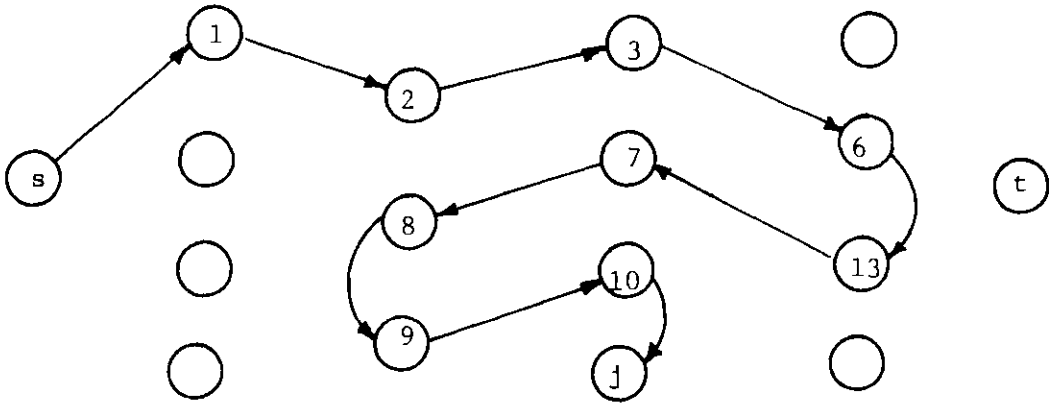
Step 1. For a given k, f^k, $\bar{N}_j^{f^k}$, j=1, ..., m use Algorithm C to find the flow augmenting path on G^{f^k}. If no such path exists, stop. The current flow f^k is optimal. Otherwise, go to Step 2.

Step 2. Change the flow along the shortest path obtained in Step

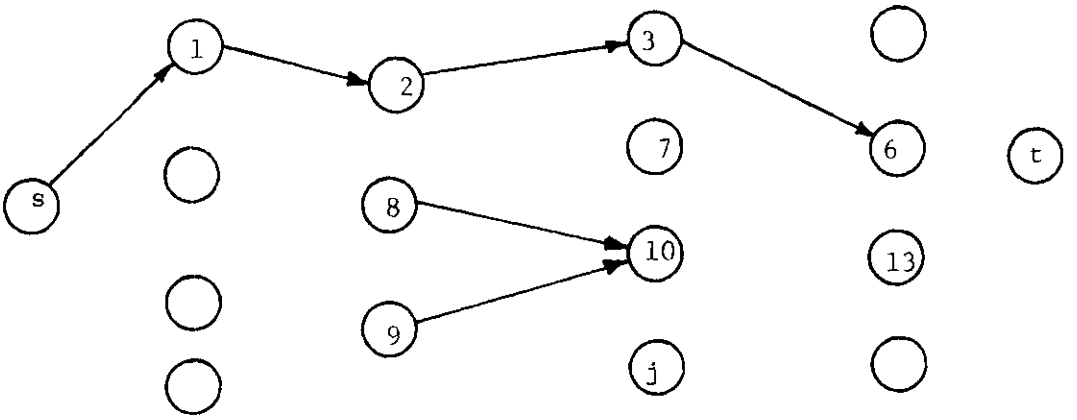
1, giving f^{k+1}. Compute the new marginal cost networks $\bar{N}_1^{f^{k+1}}$, $\bar{N}^{f^{k+1}}$, ... , $\bar{N}^{f^{k+1}}$, as described above. Go to Step 1.

Obviously, since the shortest path algorithm used in algorithm D is efficient, the whole algorithm itself is efficient. The efficiency of the algorithm is simply the efficiency of the shortest path algorithm (algorithm C) used in it.

Consider the possibility of applying Edmonds and Karp's [11] cost conversion on the G' network. The shortest distances from node s to all other nodes (except t) obtained in G' network do not necessarily represent the true distances of the original network. This fact was due to the elimination of (X_i, X_i) and (X_{i+1}, X_i) arcs from G'' network to obtain the $G'=(N',A')$ network. The following is an example of this assertion. Suppose the shortest path between s and a node $j \in N'$ is as given below



The corresponding G' network will only have the following arcs given on the G' network below



Therefore, the G' network will state that there exists no path between s and j in the original network G which contradicts the fact that there exists one. Hence, we cannot apply Edmonds and Karp's [11] minimum cost flow algorithm on the G' network since we will not be able to obtain the true shortest distances between the nodes and s and $j \neq t$. On the other hand from theorem 6 we know that all the shortest distances obtained in the $G'' = (N', A'')$ network as defined in theorem 6 correspond to the actual shortest distances on the original network $G = (N, A)$.

This observation leads us into consideration of applying Edmonds and Karp's [11] algorithm on $G'' = (N', A'')$ network. If the labels π and the flow f^k can be kept compatible on G'' network, then we can convert all the arc costs on G'' network into equivalent nonnegative ones. This, of course, will enable us to apply Dijkstra's efficient shortest path algo-

rithm to obtain the flow augmenting path.

Let G = (N,A) be decomposed into m overlapping networks $\bar{N}_1$, $\bar{N}_2$, ... , $\bar{N}_m$ as defined before. Suppose, for a given flow f, the corresponding marginal cost networks $\bar{N}_1^f$, $\bar{N}_2^f$, ... , $\bar{N}_m^f$ are as described in algorithm D. The following is the minimum cost flow algorithm which utilizes Edmonds and Karp's cost conversion on G″ network to obtain the nonnegative arc costs on G″network for flow augmentation.

Algorithm E

Initialization Step: Let $f^o = 0$ and $\pi_i^o = 0$ for all $i \epsilon N'$.

Step 1. For a given f^k, $\bar{N}_i^{f^k}$ i=1, ... , m, and d_{ij}^k, $(i,j) \epsilon A''$ apply the triple operations on each marginal cost network $\bar{N}_1^{f^k}$, $\bar{N}_2^{f^k}$ successively as described before.

Step 2. Let d_{ij}^* be the length of the shortest path from i to j $(i,j) \epsilon A''$ obtained in Step 1. Find the shortest path from s to t on G''^{f^k} with respect to the nonnegative weights.

$$\bar{d}_{ij}^k = \pi_i^k + d_{ij}^* - \pi_j^k$$

by applying Dijkstra's [9] algorithm.

If w_i^k denotes the weight of a shortest path from s to $i \epsilon N'$ in G''^{f^k} with respect to the weights $\bar{d}_{ij}^k$, set $\pi_i^{k+1} = \pi_i^k + w_i^k$; take $w_i^k = \pi_i^{k+1} = +\infty$ if the node i is inaccessible from s in G''^{f^k}. If, for some k, no flow augmenting path exists in G''^{f^k} with respect to the flow f^k, stop. Otherwise, set k = k+1. Go to Step 3.

Step 3. Compute the new marginal cost networks $\bar{N}_1^{f^k}$, ..., $\bar{N}_m^{f^k}$ with respect to the given flow f^k as described in algorithm D. Go to

Step 1.

The following theorem gives the proof of the compatibility of the labeling function π^k and flow f^k on G''^{f^k} network.

Theorem 11. For each k, f^k and π_i^k $i\epsilon N'$ are compatible. For each k and $i\epsilon N'$, π_i^k gives the weight of a shortest path from s to i in G''^{f^k} with respect to the weights d_{ij}^* and $\pi_i^{k+1} \geq \pi_i^k$.

Proof: For a general network G = (N,A), if f^k is an extreme flow, then the labeling function π^{k+1} was defined as $\pi_i^{k+1} = \pi_i^k + w_i^k$ where w_i^k is the length of the shortest path in the marginal cost network G^{f^k} from node s with respect to the weights $\bar{d}_{ij}^k$. Therefore, if P is a shortest path from u to v in G^{f^k} with respect to the weights $\bar{d}_{ij}^k$ then,

$$\sum_{(i,j)\epsilon P} \bar{d}_{ij}^k = \sum_{(i,j)\epsilon P} (\pi_i^k + d_{ij} - \pi_j^k) = \pi_u^k - \pi_v^k + \sum_{(i,j)\epsilon P} d_{ij}$$

$$= \pi_u^k - \pi_v^k + d_{uv}^* \qquad (7)$$

Where d_{uv}^* is the length of the shortest path between nodes u and v.

Consider the same network decomposed into m overlapping networks $\bar{N}_1, \ldots, \bar{N}_m$ as required by algorithm E. Suppose that nodes u and v belong to the node set N' in the network G'' = (N', A''). For any given k and f^k, the arc costs are defined as

$$\bar{d}_{ij}^k = \bar{\pi}_i^k + d_{ij}^* - \bar{\pi}_j^k$$

where d_{ij}^* is the conditional shortest path obtained in $\bar{N}_p^{f^k}$ p=1, ... , m in the first step of the algorithm. Consider the same shortest path between the nodes u and v given in G above. Since $u\epsilon N'$ and $v\epsilon N'$, from

Theorem 6 the shortest distance obtained on G'' network corresponds to the actual shortest path in the network G. Call this shortest path between u and v as P'. Hence,

$$\sum_{(i,j)\in P'} \bar{d}^k_{ij} = \sum_{(i,j)\in P'} (\bar{\pi}^k_i + d^*_{ij} - \bar{\pi}^k_j) = \bar{\pi}^k_u - \bar{\pi}^k_v + \sum_{(i,j)\in P'} d^*_{ij}$$

$$= \bar{\pi}^k_u - \bar{\pi}^k_v + d^*_{uv}$$

where d^*_{uv} is the length of the shortest path between nodes u and v. Therefore, for each pair of nodes u and v in G''^{f^k}, $\Sigma\, \bar{d}^k_{ij} = \bar{\pi}^k_u + d^*_{uv} - \bar{\pi}^k_v$ which is exactly the same as the quantity obtained in (7) for the network G^{f^k}. Therefore, if π^k_u and $\bar{\pi}^k_u$ are shown to be identical on G^{f^k} and G''^{f^k}, then the proof is complete.

The labeling function π^{k+1}, for a given flow f^k and the label π^k, for G^{f^k} was given by $\pi^{k+1}_i = \pi^k_i + w^k_i$ if node i is accessible from s in G^{f^k}. Otherwise, $\pi^{k+1}_i = w^k_i = \infty$, where w^k_i is the length of the shortest path from node s to node i in G^{f^k}. Similarly the labeling function $\bar{\pi}^{k+1}$ in G''^{f^k} network was given by:

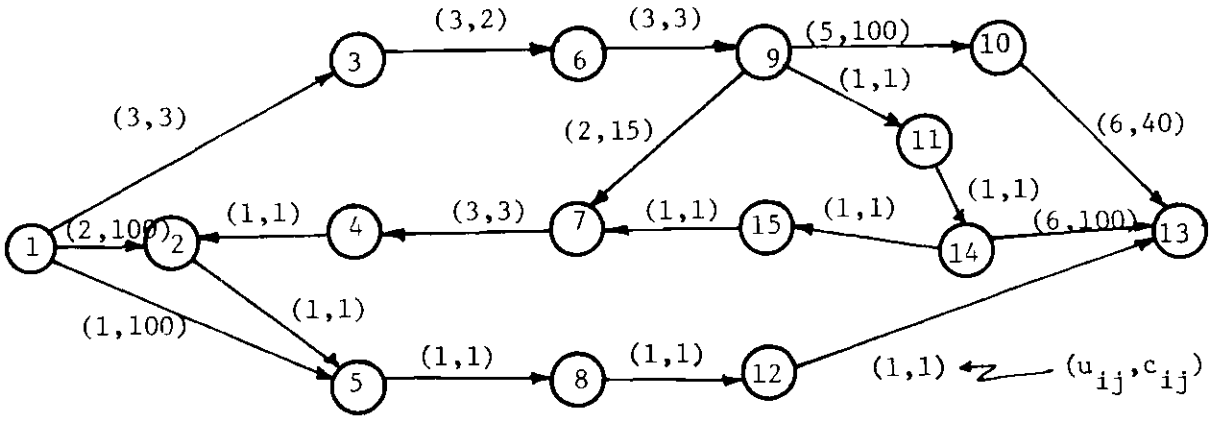
$\bar{\pi}^{k+1}_i = \bar{\pi}^k_i + \bar{w}^k_i$, if node i is accessible from node s in G''^{f^k};

$\bar{\pi}^{k+1}_i = \bar{w}^k_i = +\infty$, otherwise. The $\bar{w}^k_i$ is the length of the shortest path between nodes s and i in G''^{f^k} with respect to flow f^k. From Theorem 6 $w^k_i = \bar{w}^k_i$, and hence, $\bar{\pi}^k_i = \pi^k_i$ for all $i\in N'$, Therefore, from theorem 10, $\bar{\pi}^k_i$ and f^k are compatible in G''^{f^k} for each k, and $\bar{\pi}^k_i$ gives the length of the shortest path from node s to node i in the original network G = (N,A)

with respect to the weights d_{ij}. Also $\bar{\pi}_i^{k+1} \geq \bar{\pi}_i^{k}$. QED.

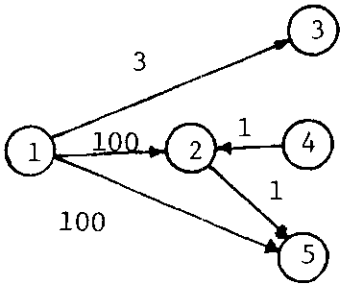
Example for Algorithm D

The example problem given for algorithm C will be utilized for the application of algorithm D. Suppose the network G = (N,A) is as given below.



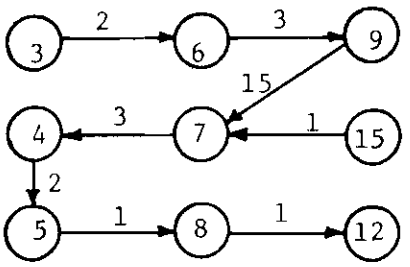
Let $N_1 = \{1,2\}$, $X_1 = \{3,4,5\}$, $N_2 = \{6,7,8\}$, $X_2 = \{9,15,12\}$, $N_3 = \{10,11,14,13\}$. Hence, $\bar{N}_1 = \{1,2,3,4,5,6,7,8,9,15,12\}$, $\bar{N}_2 = \{3,4,5,6,7,8,9,15,12\}$, $\bar{N}_3 = \{9,15,12,10,11,14,13\}$.

Iteration 1. The network $\bar{N}_1^o$ and the corresponding $D^*_{\bar{N}_1^o \bar{N}_1^o}(\bar{N}_1^o)$ are given below.



$$D^*_{\bar{N}_1^o \bar{N}_1^o}(\bar{N}_1^o) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 100 & 3 & \infty & 100 \\ \infty & 0 & \infty & \infty & 1 \\ \infty & \infty & 0 & \infty & \infty \\ \infty & 1 & \infty & 0 & 2 \\ \infty & \infty & \infty & \infty & 0 \end{bmatrix} \end{array}$$

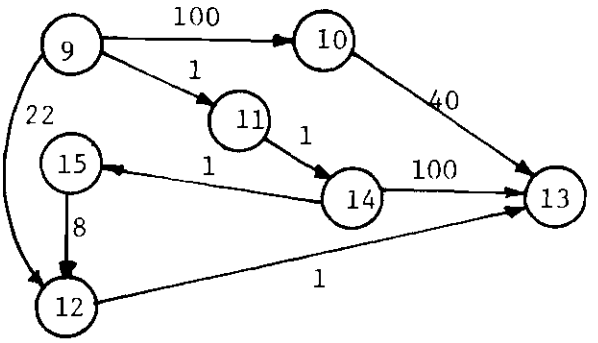
Upon replacing $d^*_{45}(\bar{N}_1^o) = 2$ with $d_{45} = \infty$ we have the following network and the corresponding conditional distance matrix for $\bar{N}_2^o$.



$D^*_{\bar{N}_2^o\bar{N}_2^o}(\bar{N}_1^o \cup \bar{N}_2^o) =$

	3	4	5	6	7	8	9	12	15
3	0	23	25	2	20	26	5	27	∞
4	∞	0	2	∞	∞	3	∞	4	∞
5	∞	∞	0	∞	∞	1	∞	2	∞
6	∞	21	23	0	18	24	3	25	∞
7	∞	3	5	∞	0	6	∞	7	∞
8	∞	∞	∞	∞	∞	0	∞	1	∞
9	∞	18	20	∞	15	21	0	22	∞
12	∞	∞	∞	∞	∞	∞	∞	0	∞
15	∞	4	6	∞	1	7	∞	8	0

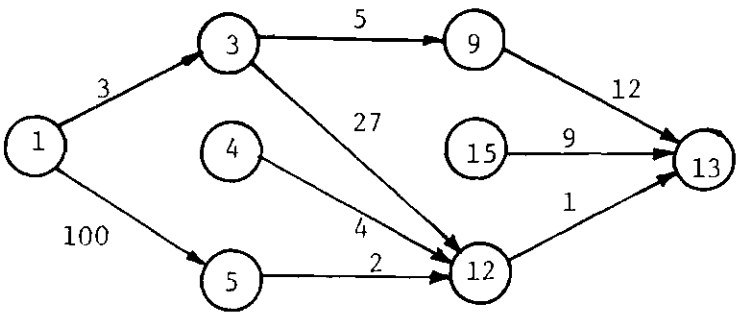
Upon replacing $d_{9,12} = \infty$ by $d^*_{9,12}(\bar{N}_1^o \cup \bar{N}_2^o) = 22$ and $d_{15,12} = \infty$ by $d^*_{15,12}(\bar{N}_1^o \cup \bar{N}_2^o) = 8$ we get the following network and corresponding shortest distance matrix $D^*_{\bar{N}_3^o\bar{N}_3^o}(N)$.



$$D^{*}_{\bar{N}_3\bar{N}_3}(N) =$$

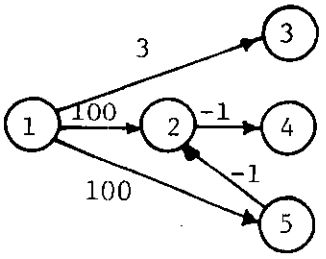
	9	10	11	12	13	14	15
9	0	100	1	11	12	2	3
10	∞	0	∞	∞	40	∞	∞
11	∞	∞	0	10	11	1	2
12	∞	∞	∞	0	1	∞	∞
13	∞	∞	∞	∞	0	∞	∞
14	∞	∞	∞	9	10	0	1
15	∞	∞	∞	8	9	∞	0

Now we can form $G' = (N', A')$ network.



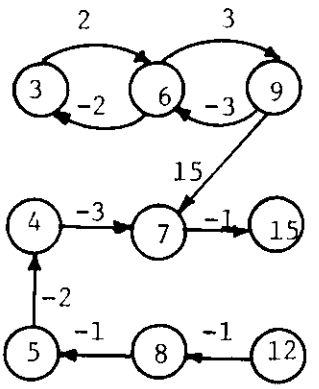
By using algorithm B we obtain $w_1^o = 0$, $w_3^o = 3$, $w_4^o = \infty$, $w_5^o = 100$, $w_9^o = 8$, $w_{15}^o = \infty$, $w_{12}^o = 30$ and $w_{13}^o = 20$. Therefore, the flow augmenting path is 1-3-9-13 with length 20 which corresponds to the actual shortest path 1-3-9-11-14-15-7-4-2-5-8-12-13. The maximum flow which can be sent along this path is 1. Therefore, $f' = f^o + 1 = 1$, $f_{13} = 1$, $f_{39} = 1$, $f_{9,11} = 1$, $f_{11,14} = 1$, $f_{14,15} = 1$, $f_{15,7} = 1$, $f_{7,4} = 1$, $f_{42} = 1$, $f_{25} = 1$, $f_{5,8} = 1$, $f_{8,12} = 1$ and $f_{12,13} = 1$. The marginal cost networks $\bar{N}^1$, $\bar{N}_2^1$, . . . , $\bar{N}_m^1$ and their respective shortest distances are given in iteration 2.

Iteration 2. $\bar{N}_1^1$ and its corresponding conditional shortest distance matrix is given below.



$$D^*_{\bar{N}_1^1}(\bar{N}_1^1) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 99 & 3 & 98 & 100 \\ \infty & 0 & \infty & -1 & \infty \\ \infty & \infty & 0 & \infty & \infty \\ \\ \infty & \infty & \infty & 0 & \infty \\ \infty & -1 & \infty & -2 & 0 \end{bmatrix} \end{array}$$

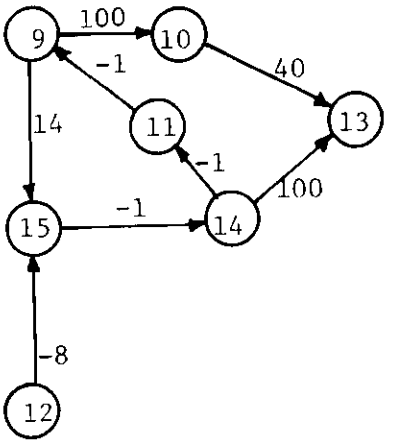
Upon replacing $d_{54} = \infty$ by $d^*_{54}(\bar{N}_1^1) = -2$ we get the following marginal cost network, $\bar{N}_2^1$ and its corresponding conditional shortest distance matrix.



$$D^{*}_{\bar{N}_2^{-1}\bar{N}_2^{-1}}(\bar{N}_1^{-1} \cup \bar{N}_2^{-1}) =$$

	3	4	5	9	15	12
3	0	∞	∞	5	19	∞
4	∞	0	∞	∞	-4	∞
5	∞	-2	0	∞	-6	∞
.	.	.	. .	.	.	.
.	.	.	. .	.	.	.
.	.	.	. .	.	.	.
.	.	.	. .	.	.	.
.	.	.	. .	.	.	.
9	-5	∞	∞	0	14	∞
15	∞	∞	∞	∞	0	∞
12	∞	-4	-2	∞	-8	∞

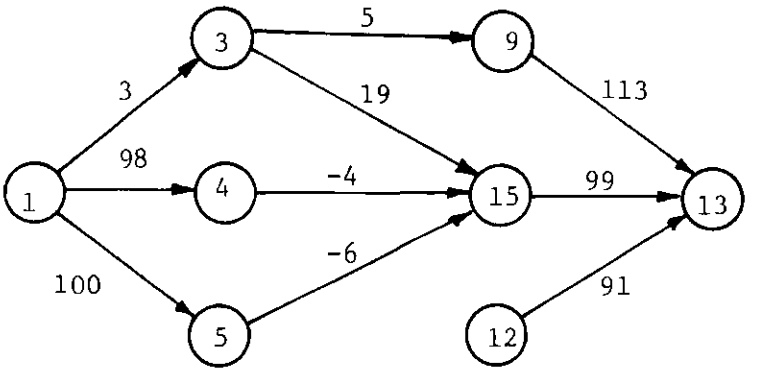
Upon replacing $d_{9,15} = \infty$ by $d^{*}_{9,15} = 14$ and $d_{12,15}$ by $d^{*}_{12,15} = -8$ we obtain the following marginal cost network $\bar{N}_3^{1}$ and its corresponding shortest distance matrix $D_{\bar{N}_3\bar{N}_3}(N)$.



$D^{*}_{\bar{N}_3\bar{N}_3}(N)$ =

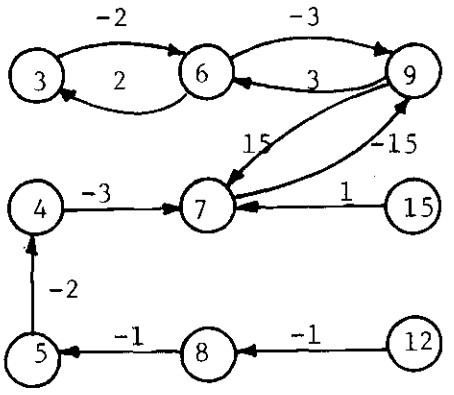
	9	15	12	13	. . . .
9	0	14	∞	113	. . . .
15	-3	0	∞	99	. . . .
12	-11	-8	0	91	. . . .
13	∞	∞	∞	∞	. . . .
.	.	.	.	.	. . . .
.	.	.	.	.	. . . .
.	.	.	.	.	. . . .

Following is the G'^{f^1} network.



Application of algorithm B will give the following labels: $w_1^1 = 0$, $w_3^1 = 3$, $w_4^1 = 98$, $w_5^1 = 100$, $w_9^1 = 8$, $w_{15}^1 = \min\{w_3^1 + 19, w_4^1 - 4, w_5^1 - 6\} = w_3^1 + 19 = 22$, $w_{12}^1 = \infty$, $w_{13}^1 = \min\{w_9^1 + 113, w_{15}^1 + 99, w_{12}^1 + 91\} = w_9^1 + 113 = 121$. Therefore, the flow augmenting path is 1-3-9-13 with length 121. This path corresponds to the actual path 1-3-6-9-7-15-14-13. The maximum flow that can be sent along this path is 1. Hence $f^2 = f^1 + 1 = 1+1 = 2$, $f_{13} = 2$, $f_{36} = 2$, $f_{69} = 2$, $f_{97} = 1$, $f_{15,7} = 0$, $f_{14,15} = 0$, $f_{14,13} = 1$, $f_{7,9} = 1$, $f_{4,2} = 1$, $f_{5,8} = 1$, $f_{8,12} = 1$, $f_{12,13} = 1$ and all other $f_{ij} = 0$.

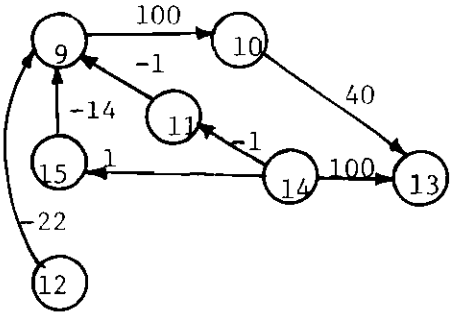
Iteration 3. The marginal cost network $\bar{N}_1^2$ and corresponding shortest distance matrix is exactly the same as the marginal cost network $\bar{N}_1^1$ and its corresponding shortest distance matrix given in iteration 1. Upon replacing the original distances in $\bar{N}_2^2$ by the conditional distances obtained in $\bar{N}_1^2$ we have the following network and its corresponding D^* matrix:



$$D^{*\bar{2}}_{\bar{N}_2\bar{N}_2}(\bar{N}_1 \cup \bar{N}_2) =$$

	3	4	5	9	15	12
3	0	∞	∞	5	∞	∞
4	-23	0	∞	-18	∞	∞
5	-25	-2	0	-20	∞	∞
.	.	.	. .....	.	.	.
.	.	.	. .....	.	.	.
.	.	.	. .....	.	.	.
9	-5	∞	∞	0	∞	∞
15	-19	∞	∞	-14	0	∞
12	-27	-4	-2	-22	∞	0

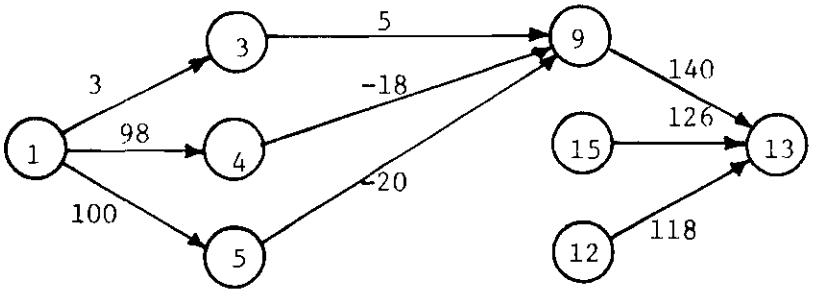
Replacing $d_{15,9} = -14$ and $d_{12,9} = \infty$, by $d^*_{12,9} = -22$ we get the marginal cost network $\bar{N}_3^{\bar{2}}$ and its corresponding shortest distance matrix.



$D^{*\bar{2}}_{\bar{N}_3\bar{N}_3}(N)$

	9	15	12	13 ....
9	0	∞	∞	140 ....
15	-14	0	∞	126 ....
12	-22	∞	0	118 ....
13	∞	∞	∞	0 ....
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

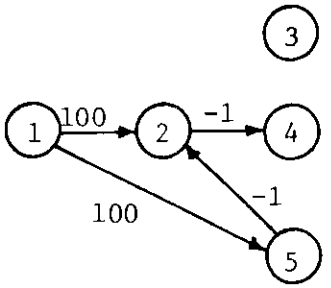
G'^{f^2} is given below.



From the application of algorithm B we get the following labels: $w_1^2 = 0$, $w_3^2 = 3$, $w_4^2 = 98$, $w_5^2 = 100$, $w_9^2 = 8$, $w_{15}^2 = \infty$, $w_{13}^2 = 148$. The shortest path is 1-3-9-13 with the length 148 which corresponds to the actual shortest path 1-3-6-9-10-13 in the marginal cost network of the original network, G^{f^2}. The maximum flow that can be sent along this path is 1. Hence, $f^3 = f^2+1 = 2+1 = 3$, $f_{13} = 3$, $f_{36} = 3$, $f_{69} = 3$, $f_{9,10} = 1$, $f_{10,13} = 1$ and all other flow as obtained in iteration 2 before.

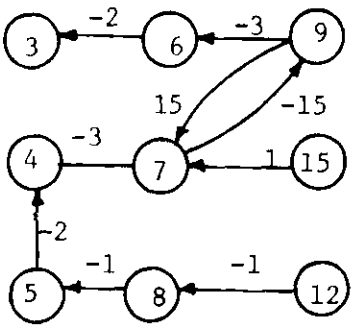
Iteration 4.

$\bar{N}_1^3$:



$$D^*_{\bar{N}_1^3 \bar{N}_1^3}(\bar{N}_1^3) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 99 & \infty & 98 & 100 \\ \infty & 0 & \infty & -1 & \infty \\ \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty \\ \infty & -1 & \infty & -2 & 0 \end{bmatrix} \end{array}$$

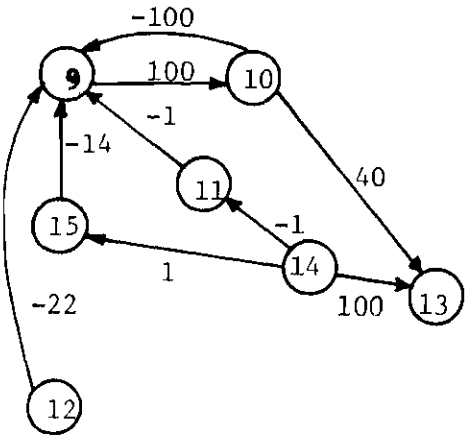
$\bar{N}_2^3$:



$$D^{*}_{\bar{N}_2^3 \bar{N}_2^3}(\bar{N}_1^3 \cup \bar{N}_2^3) =$$

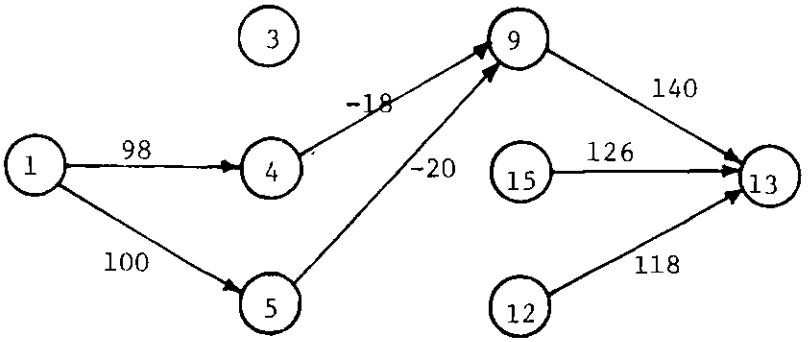
	3	4	5	...	9	15	12
3	0	∞	∞	...	∞	∞	∞
4	-23	0	∞	...	-18	∞	∞
5	-25	-2	0	...	-20	∞	∞
.	.	.	.	...	.	.	.
.	.	.	.	...	.	.	.
.	.	.	.	...	.	.	.
9	-5	∞	∞	...	0	∞	∞
15	-19	∞	∞	...	-14	0	∞
12	-27	-4	-2	...	-22	∞	0

$\bar{N}_3^3$:



$$D^{*}_{N_3 N_3}(N) = \begin{array}{c} \\ 9 \\ 15 \\ 12 \\ 13 \\ \cdot \\ \cdot \\ \cdot \end{array} \begin{array}{c} \begin{array}{cccc} 9 & 15 & 12 & 13 & \cdots. \end{array} \\ \begin{bmatrix} 0 & \infty & \infty & 140 & \cdots. \\ -14 & 0 & \infty & 126 & \cdots. \\ -22 & \infty & 0 & 118 & \cdots. \\ \infty & \infty & \infty & 0 & \cdots. \\ \cdot & \cdot & \cdot & \cdot & \cdots. \\ \cdot & \cdot & \cdot & \cdot & \cdots. \\ \cdot & \cdot & \cdot & \cdot & \cdots. \end{bmatrix} \end{array}$$

The corresponding G'^{f^3} network is given below.

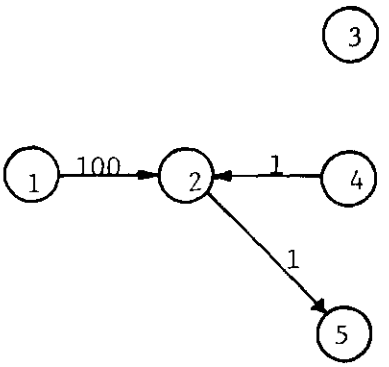


Application of algorithm B on G'^{f^3} gives the following labels: $w_1^3 = 0$, $w_3^3 = \infty$, $w_4^3 = 98$, $w_9^3 = 80$, $w_{15}^3 = w_{12}^3 = \infty$ and $w_{13}^3 = 220$. The shortest path is 1-4-9-13 or 1-5-9-13, both of them corresponding to the same shortest path 1-5-2-4-7-9-10-13 in the marginal cost network G^{f^3}. The maximum flow which can be sent along this path is 1. Hence, $f^4 = f^3 + 1 = 3+1 = 4$, $f_{15} = 1$, $f_{25} = 0$, $f_{42} = 0$, $f_{74} = 0$, $f_{97} = 0$, $f_{9,10} = 2$, $f_{10,13} = 0$ and all other arc flows equal to the values given in itera-

ation 3.

Iteration 5.

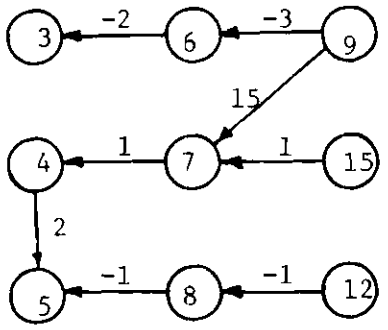
$\bar{N}_1^4$:



$$D^*_{\bar{N}_1^4 \bar{N}_1^4}(\bar{N}_1^4) = 3$$

	1	2	3	4	5
1	0	100	∞	∞	101
2	∞	0	∞	∞	1
3	∞	∞	0	∞	∞
4	∞	∞	∞	0	2
5	∞	∞	∞	∞	0

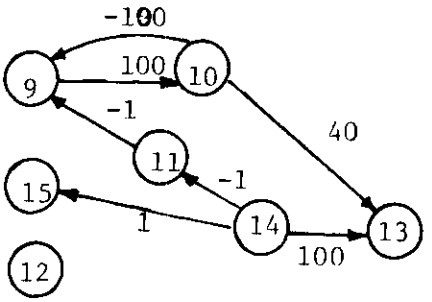
$\bar{N}_2^4$:



$$D^*_{\bar{N}_2^4 \bar{N}_2^4}(\bar{N}_1^4 \cup \bar{N}_2^4) =$$

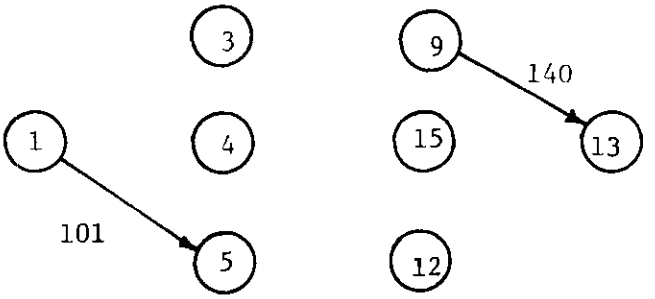
	3	4	5	...	9	15	12
3	0	∞	∞	...	∞	∞	∞
4	∞	0	2	...	∞	∞	∞
5	∞	∞	0	...	∞	∞	∞
.	.	.	.	...	.	.	.
.	.	.	.	...	.	.	.
.	.	.	.	...	.	.	.
9	-5	16	18	...	0	∞	∞
15	∞	2	4	...	∞	0	∞
12	∞	∞	-2	...	∞	∞	0

Obviously, we can stop the algorithm here, since no path exists between $X_1 = \{3,4,5\}$ and $X_2 = \{9,15,12\}$. We will continue this process to show that no more flow augmenting path exists in G' network either.

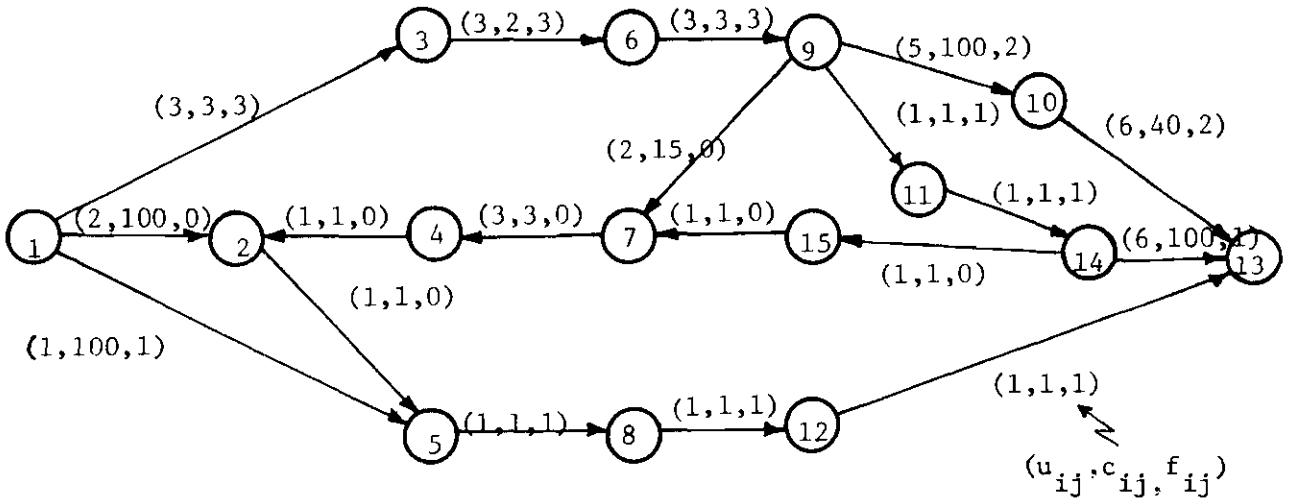


$$D^{*}_{N_3 N_3}(N) = \begin{array}{c} \\ 9 \\ 12 \\ 15 \\ . \\ . \\ . \end{array} \begin{array}{c} 13 \\ \begin{bmatrix} 140 & \cdots\cdots\cdots \\ \infty & \cdots\cdots\cdots \\ \infty & \cdots\cdots\cdots \\ . & \cdots\cdots\cdots \\ . & \cdots\cdots\cdots \\ . & \cdots\cdots\cdots \end{bmatrix} \end{array}$$

The marginal cost G'^{f^4} will be as follows:



There exists no flow augmenting path in G'^{f^4} with respect to the flow f^4. Therefore, the flow f^4 and the arc flows f_{ij} obtained in iteration 4 are optimal. The network G and the corresponding minimum cost maximun flow is depicted below.



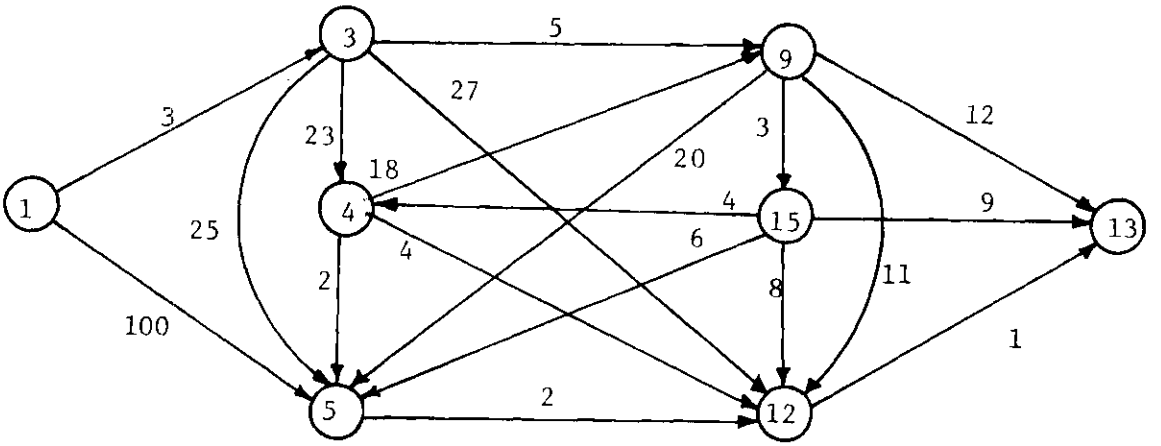
Example for Algorithm E

The same network G = (N,A), given in the previous example for the application of algorithm D, will be used for the application of algorithm E, as well.

Initialization: Let $\pi_1^o = \pi_3^o = \pi_4^o = \pi_5^o = \pi_9^o = \pi_{15}^o = \pi_{12}^o$

$= \pi_{13}^o = 0$ and $f^o = 0$.

Iteration 1. $\bar{N}_1^o$, $D^*_{\bar{N}_1^o \bar{N}_1^o}(\bar{N}_1^o)$, $\bar{N}_2^o$, $D^*_{\bar{N}_2^o \bar{N}_2^o}(\bar{N}_1^o \cup \bar{N}_2^o)$, $\bar{N}_3^o$ and $D^*_{\bar{N}_3^o \bar{N}_3^o}(N^o)$ are exactly as the values obtained in iteration 1 of the previous example. Upon constructing the G''^{f^o} network, we get the following graph.



Since $\pi_i^o = 0$ for all i,

$$\bar{d}_{ij}^o = \pi_i^o + d_{ij}^* - \pi_j^o = d_{ij}^o .$$

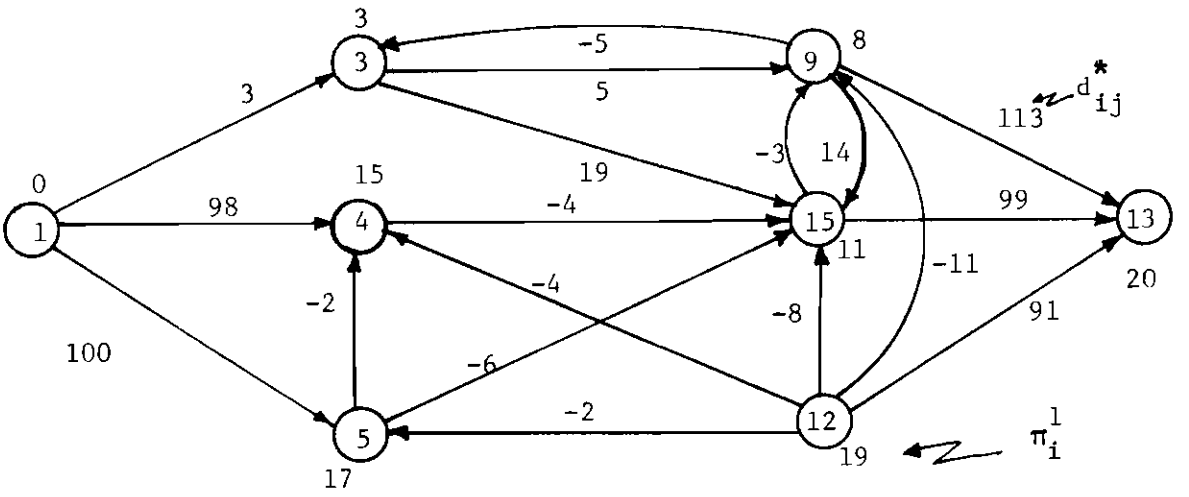
Application of Dijkstra's [9] algorithm on the G''^{f^o} network will give $w_1^o = 0$, $w_3^o = 3$, $w_4^o = 15$, $w_5^o = 17$, $w_9^o = 8$, $w_{15}^o = 11$, $w_{12}^o = 19$, $w_{13}^o = 20$. There are several shortest paths in G''^{f^o}. We will choose the one with the minimum number of arcs in it (note that all those paths correspond to the same path in G). The actual path corresponding to the shortest path 1-3-9-13 in G''^{f^o} is the path 1-3-6-9-11-14-15-7-4-2-5-8-12-13. This is the flow augmenting path obtained in iteration 1 of the previous examples. Therefore, the flow changes will be exactly as given in iteration 1 of the previous example with $f' = 1$.

From the formula $\pi_i^{k+1} = \pi_i^k + w_i^k$ we update the labels on the nodes of G'' as follows:

$\pi_1^1 = 0 + 0 = 0$, $\pi_3^1 = 0 + 3 = 3$. $\pi_4^1 = 0 + 15 = 15$, $\pi_5^1 = 0 + 17 = 17$, $\pi_9^1 = 0 + 8 = 8$, $\pi_{15}^1 = 0 + 11 = 11$, $\pi_{12}^1 = 0 + 19 = 19$, $\pi_{13}^1 = 0 + 20 =$

20.

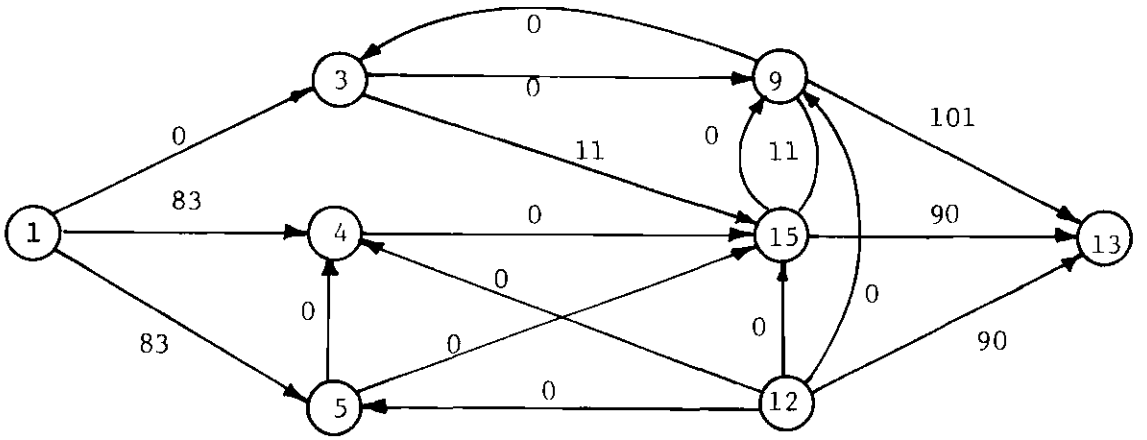
Iteration 2. Since we have used the same flow augmenting path as in iteration 1 of the previous example, the conditional shortest distances for $\bar{N}_1^1$, $\bar{N}_2^1$ and $\bar{N}_3^1$ will be the same as in iteration 2 of the previous example. From the shortest distance tables in iteration 2 of the previous example we can construct G''^{f^1} network as given below.



We can compute the nonnegative cost coefficients for each arc by the formula:

$$\bar{d}_{ij}^1 = \pi_i^1 + d^*_{ij} - \pi_j^1$$

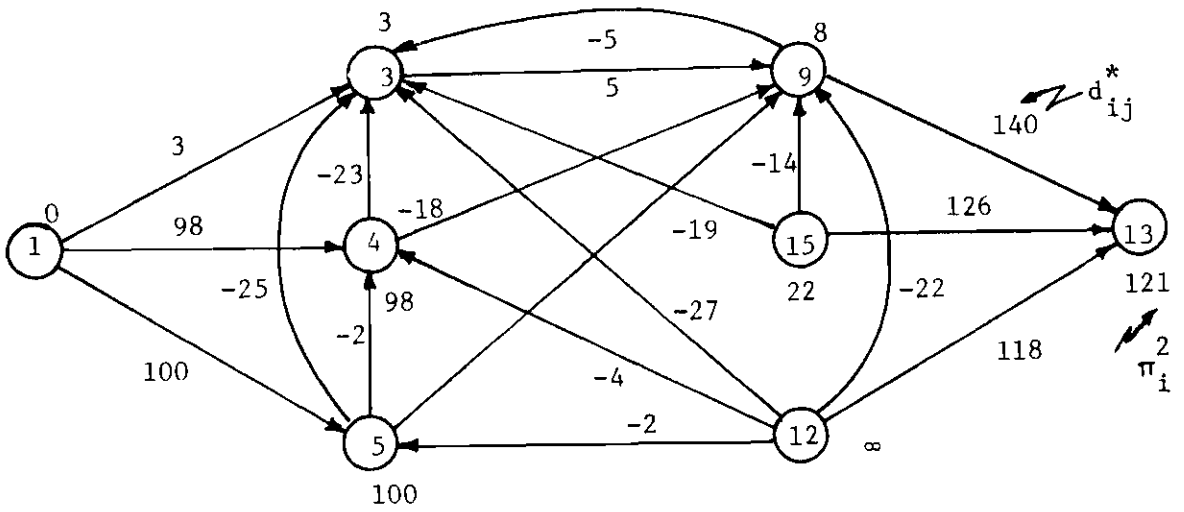
Therefore, $\bar{d}_{13}^1 = 0 + 3 - 3 = 0$, $\bar{d}_{14}^1 = 0 + 98 - 15 = 83$, $\bar{d}_{15}^1 = 0 + 100 - 17 = 83$, $\bar{d}_{54}^1 = 0$, $\bar{d}_{39}^1 = 0$, $\bar{d}_{3,15}^1 = 0$, $\bar{d}_{4,15}^1 = 0$, $\bar{d}_{5,15}^1 = 0$, etc., ... The nonnegative cost G''^{f^1} is given below.



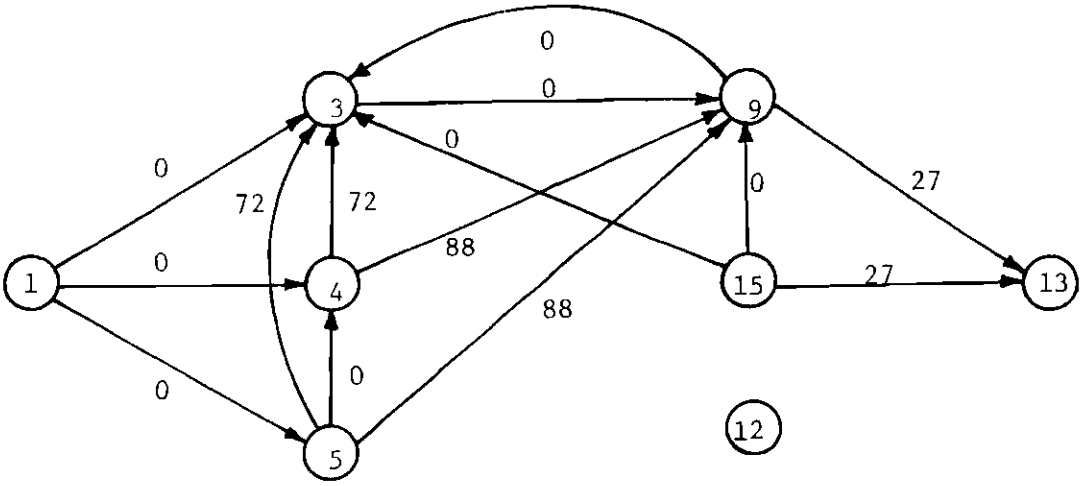
From the application of Dijkstra's algorithm we get $w_1^1 = 0$, $w_3^1 = 0$, $w_4^1 = 83$, $w_9^1 = 0$, $w_{15}^1 = 5$, $w_{12}^1 = \infty$, $w_{13}^1 = 101$, $w_5^1 = 83$.

Among the several shortest paths in G''^{f^1}, we will choose the minimum arc shortest path 1-3-9-13 which corresponds to the path 1-3-6-9-7-15-14-13 of the marginal cost network of the original network. Note that this path is exactly the same as the one obtained at the end of iteration 2 of the previous example. Therefore, $f^2 = f^1 + 1 = 2$ and the arc flows are as given in iteration 2 of the previous example. Updating the labeling function by the formula $\pi_i^{k+1} = \pi_i^k + w_i^k$ will yield $\pi_1^2 = 0$, $\pi_3^2 = 3$, $\pi_4^2 = 98$, $\pi_5^2 = 100$, $\pi_9^2 = 8$, $\pi_{15}^2 = 22$, $\pi_{12}^2 = \infty$, $\pi_{13}^2 = 121$.

Iteration 3. Again, since we used the same flow augmenting path, we can construct G''^{f^2} network from the tables of iteration of the previous example. The arc costs and the node labels are shown on the network.



Using the formula $\tilde{d}^2_{ij} = \pi^2_i + d^*_{ij} - \pi^2_j$ we obtain the nonnegative arc costs as given below.

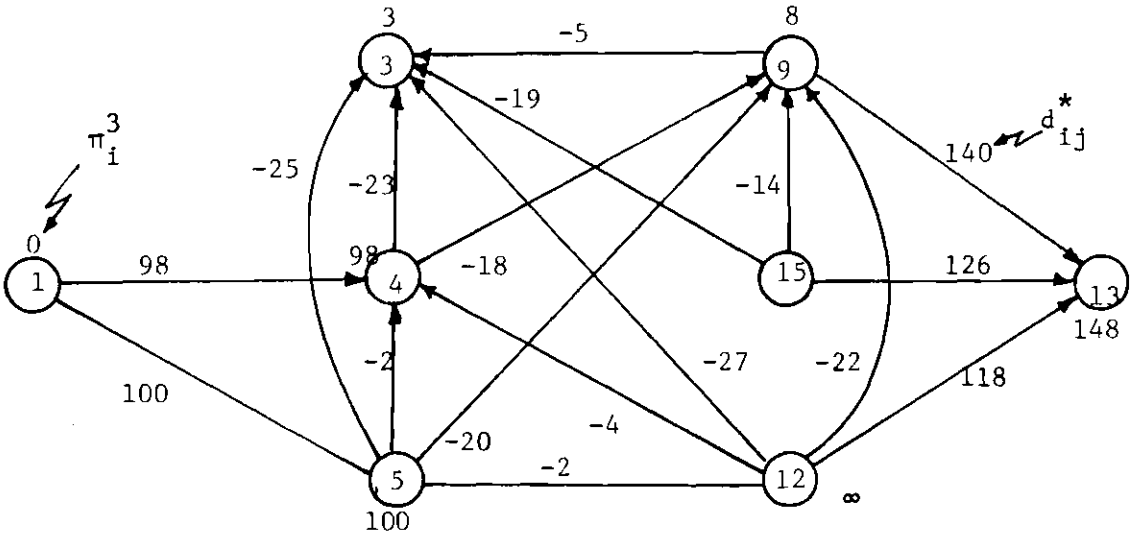


Application of Dijkstra's algorithm gives $w^2_1 = 0$, $w^2_3 = 0$, $w^2_4 = 0$, $w^2_5 = 0$, $w^2_9 = 0$, $w^2_{15} = \infty$, $w^2_{12} = \infty$, $w^2_{13} = 27$. The shortest path 1-3-9-13 corresponds to the flow augmenting path 1-3-6-9-10-13 in the original network which is exactly the same path obtained in iteration 3 of the previous example. Therefore, $f^3 = f^2 + 1 = 3$ and f_{ij}'s are given in

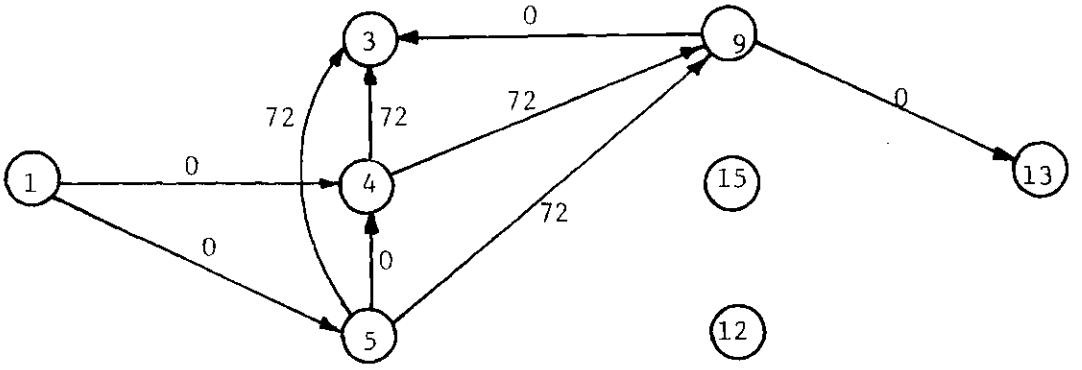
iteration 3 of the previous example. From $\pi_i^3 = \pi_i^2 + w_i^2$, the updated labels will be as follows:

$\pi_1^3 = 0$, $\pi_3^3 = 3$, $\pi_4^3 = 98$, $\pi_5^3 = 100$, $\pi_9^3 = 8$, $\pi_{15}^3 = \infty$, $\pi_{12}^3 = \infty$, $\pi_{13}^3 = 148$.

Iteration 4. With a similar argument, the $G''f^3$ network can be constructed from the tables of iteration 4 of the previous example as given below.



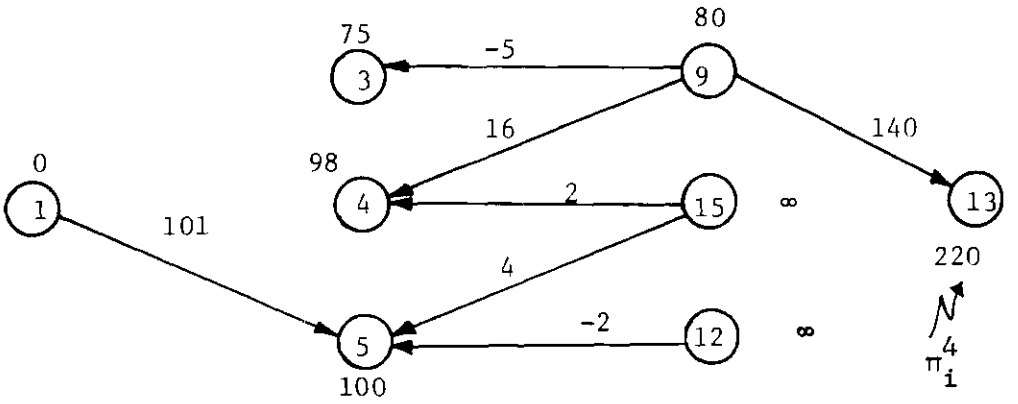
The equivalent nonnegative cost network is obtained from $\bar{d}_{ij}^3 = \pi_i^3 + d_{ij}^* - \pi_j^3$ and depicted below.



From Dijkstra's algorithm we obtain $w_1^3 = 0$, $w_3^3 = 72$, $w_4^3 = 0$, $w_5^3 = 0$, $w_9^3 = 72$, $w_{15}^3 = \infty$, $w_{12}^3 = \infty$, $w_{13}^3 = 72$. The shortest paths 1-4-9-13 or 1-5-9-13 or 1-5-2-4-7-9-10-13 in the marginal cost network of the original network. Again, this path is the same as the path obtained in iteration 4 of the previous example. Hence, $f^4 = f^3 + 1 = 4$, and all the arc flows are as given in the corresponding iteration of the previous example.

The updated labels are given as: $\pi_1^4 = 0$, $\pi_3^4 = 75$, $\pi_4^4 = 98$, $\pi_5^4 = 100$ $\pi_9^4 = 80$, $\pi_{15}^4 = \infty$, $\pi_{12}^4 = \infty$, $\pi_{13}^4 = 220$.

Iteration 5. Since we are using the same flow augmenting paths as in the previous example, we can construct G''^{f^4} as given below.



No flow augmenting path exists in G''^{f^4} with respect to the flow f^4. Therefore, the flow f^4 obtained at iteration 4 is optimal. This flow is exactly the same flow obtained in the previous example.

Theoretical Efficiency

As discussed earlier, the efficiency of a flow augmenting algorithm is a linear function of the efficiency of the shortest path algorithm employed in it. The efficiency of the shortest path algorithm (Algorithm C) employed in algorithm D is given Chapter IV as

$$mu^3 + (6m-6)\ u^2v + (12m-8)\ uv^2 + (8m-14)\ v^3 + (m-2)\ v^2 + v.$$

Therefore, algorithm D requires $0(mu^3)$ additions and comparisons for determining a flow augmenting path at each iteration.

The number of additions and comparisons required by algorithm E for determining a flow augmenting path can be computed as:

1) $(m-2)(u+2v)^3 + 2(u+v)^3$ additions and comparisons for determining the conditional shortest distances on $\bar{N}_i^k$, i=1,...,m.

2) $(6m-14)\ v^2 + (2m-10)\ v$ additions for obtaining the nonnegative arc weights in G''^{f^k}.

3) $[(m-1)\ v + 2]^2$ additions and comparisons for determing the shortest distances w_i^k on the G''^{f^k} network.

4) $[(m-1)\ v + 1]$ additions and comparisons for updating the labeling function π^{k+1}.

Therefore, algorithm E reaches a flow augmenting path in

$$mu^3 + (6m-6)\ u^2v + (12m-8)\ uv^2 + (8m-14)\ v^3 + (m^2 + 4m - 13)\ v^2 + (3m + 6)\ v + 5 \text{ additions and}$$

$$mu^3 + (6m-6)\ u^2v + (12m-8)\ uv^2 + (8m-14)\ v^3 + (m^2 - 2m + 1)\ v^2 + (5m-5)\ v + 5 \text{ comparisons.}$$

For large values of m and u and small values of v the ratio of the number of operations in algorighm D to the number of operations in algorithm E can be approximated by using their higher order terms as

$$\frac{mu^3}{mu^3} = 1.$$

This implies that algorithm E is as efficient as algorithm D in its theoretical upper bound. Theoretically, algorithm D is expected to take a smaller amount of time at each flow augmentation than algorithm E, but for $u >> v$ this difference becomes less and less significant. If the optimum flow requires k flow augmentations then the number of addi-

tions and comparisons saved by algorithm D over algorithm E can be approximated by $k[m^2v^2]$. As an example, if 1000 flow augmentations are needed to reach the minimum cost flow in a given network, and m = 10, v = 20 and u = 100, then the amount of additions and comparisons saved by algorithm D over algorithm E is $1000[(10)^2(20)^2] = 40,000,000$ additions and comparisons. The approximate number of additions and comparisons required by algorithm D is

$$1000[10(100)^3] = 10,000,000,000.$$

Therefore, the efficiency of algorithm D is only .4% better than the efficiency of algorithm E. Thus, we can conclude that both algorithm D and algorithm E are efficient and good algorithms.

CHAPTER VI

DISCUSSION

In the shortest path algorithm (Algorithm C), the conditional shortest distances $D_{\bar{N}_i \bar{N}_i}(\bar{N}_1 \cup \ldots \cup \bar{N}_i)$ i=1, . . ., m are determined by utilizing Floyd's matrix algorithm. Therefore, it requires $(u+v)^3$ additions and comparisons to obtain the conditional distances in the first and last submatrices, whereas it needs $(u+2v)^3$ additions and comparisons for each $D^{*}_{\bar{N}_i \bar{N}_i}$, i=2, ..., m-1. Obviously, the higher order term mu^3 in the total number of additions and comparisons in Algorithm C is due to this matrix operation. Again, we pay the price of the extra information obtained in the matrix method which is not being used.

The discussion in this section will deal with the possibility of having another method which can supply the required information in $O(kmu)^2$ additions and comparisons, where $k < u$. Observe that, if f is an extreme flow in G = (N,A), then f is also extreme in $\bar{N}_i$, i=1, ..., m. This can easily be shown by stating that G^f does not contain any negative cycles with respect to f. From this observation, we can conclude that for a given extreme flow f in G, there exists a labeling function π in $\bar{N}^f_k$, k=1, ..., m such that for each $(i,j) \in \bar{N}^f_k$, k=1, ..., m

$$\pi_i + d_{ij} - \pi_j \geq 0.$$

Therefore, we can find a set of π's for a given f in $\bar{N}^f_k$ such that π and

f are compatible in $\bar{N}_k$.

Let f and π be compatible in $\bar{N}_k$. Define the nonnegative arc costs as $\bar{d}_{ij} = \pi_i + d_{ij} - \pi_j$. Let P be a path between $u^* \epsilon X_{k-1}$ and $v^* \epsilon X_k$. Then P is a shortest path from node u^* to node v^* with respect to the arc weights d_{ij} if, and only if, P is a shortest path from node u^* to v^* with respect to the nonnegative arc weights $\bar{d}_{ij}$.

In order to be able to find a compatible set of π's for given f, we need to define the sink and the source. Remember that π's are updated by using the equation $\pi_i^{k+1} = \pi_i^k + w_i^k$, where w_i^k is the length of the shortest path of asource to a node i. This observation implies the impossibility of using one set of π's for each subnetwork k to determine $D^*_{X_{k-1}X_{k-1}}$, $D^*_{X_{k-1}X_k}$, $D^*_{X_kX_{k-1}}$ and $D^*_{X_kX_k}$. By letting $u^* \epsilon X_{k-1}$ and $v^* \epsilon X_k$ as the source and the sink of the network $\bar{N}_k$ we can find a set of labeling function π which is compatible for a given extreme flow f in $\bar{N}_k$. Therefore, the shortest path problem can be solved on the nonnegative arc costs $\bar{d}_{ij} = \pi_j + d_{ij} - \pi_j$. Dijkstra's algorithm will give us $D^*_{u^*X_{k-1}}$ as well as $D^*_{u^*X_k}$. Therefore, in order to find $D^*_{X_{k-1}X_{k-1}}$ and $D^*_{X_{k-1}X_k}$ each one of the nodes $u^* \epsilon X_{k-1}$ have to be made a source once (the sink v^* being any node in X_k). Similarly, in order to find the conditional shortest distances $D^*_{X_kX_k}$ as well as $D^*_{X_kX_{k-1}}$, we need to make each $v^* \epsilon X_k$ a souce once (sink being any arbitrary node u^* in X_{k-1}). Thus, we need to carry v+v = 2v labeling functions in order to determine the conditional shortest distances $D^*_{X_{k-1}X_{k-1}}$, $D^*_{X_{k-1}X_k}$, $D^*_{X_kX_{k-1}}$ and $D^*_{X_kX_k}$ in a subnetwork $\bar{N}_k$, k=2, ..., m-1. For the first network, we need (v+1) labeling functions and for the last subnetwork only one labeling will suffice (by labeling from the last node).

The number of times Dijkstra's algorithm needs to be applied on the subnetworks can be given as:

v+1 times on the first subnetwork

2v times on subnetworks $\bar{N}_2, \ldots, \bar{N}_{m-1}$

1 time on subnetwork $\bar{N}_m$

Since Dijkstra's algorithm requires $n^2/2$ additions and n^2 comparisons for an n-node network, it will require

$$1/2\left\{(v+1)(u+v)^2 + (m-2)(u+2v)^2\ 2v + (u+v)^2\right\} = Q$$

additions and 2Q comparisons in order to be able to construct G′ network. Finding the shortest path on G′ requires $(m-2)\ v^2 + v$ additions and $(m-2)\ v^2 + v$ comparisons. Thus, the total number of additions and comparisons required for each flow augmentation is given by

$Q + (m-2)\ v^2 + v$ additions and

$2Q + (m-2)\ v^2 + v$ comparisons.

These quantities can be rewritten as

$$1/2\left\{(2m-3)\ u^2v + (8m-14)\ uv^2 + (8m-15)\ v^3 + 2u^2 + (2m-2)\ v^2 + 4uv + 2v\right\}$$ additions and

$$(2m-3)\ u^2v + (8m-14)\ uv^2 + (8m-15)\ v^3 + 2u^2 + mv^2 + 4uv + v$$ comparisons.

Therefore, the efficiency of each flow augmentation is increased approximately by the ratios $(u+2v)/v$ in the number of additions and $(\frac{u+2v}{2v})$ in the number of comparisons. The new refinement will change the order

$0(mu^3)$ of the suggested algorithm into $0(mu^2v)$ additions and $0(2mu^2v)$ comparisons.

The sacrifice obviously, in this modification, is the requirement of the extra storage for $[(2v) \times (u+2v)]$ matrices to store the labels. In addition, we will need $2(u+v)^2(v+2) + 2(m-2)(u+2v)^2 2v$ additions for updating the costs and $(m-2)(u+2v) + 2(u+v)$ additions and comparisons for updating the labeling functions π in the subnetworks. If these extra additions and comparisons are included in the upper bound obtained, the proposed modification will require

$$1/2\{(10m-15)\ u^2v + (40m-70)\ uv^2 + (40m-75)\ v^3 + 10u^2 + (2m+6)\ v^2 + 16uv + (4m-2)\ v + 2mu\}$$ additions and

$$(2m-3)\ u^2v + (8m=14)\ uv^2 + (8m-15)\ v^3 + 2u^2 + mv^2 + 4uv + mu + (2m-1)\ v$$ comparisons.

The higher order term in this number is still $0(5mu^2v)$ compared to $0(mu^3)$ in the previous algorithm. Approximately for $u > 5u$ the proposed modification reaches the flow augmenting path faster.

CHAPTER VII

CONCLUSIONS AND RECOMMENDATIONS

Although the theoretical upper bound obtained for Algorithm A (min-cut algorithm) is quadratically more efficient than a nodecomposition labeling algorithm, we know in practice that the theoretical upper bounds are seldom reached. As pointed out in Chapter II earlier, Algorithm A has double superiority over a no-decomposition algorithm. The first superiority comes from the lower theoretical upper bound in the number of flow augmentations. The second superiority comes from the fact that the flow augmentation in Algorithm A is always performed on a smaller size network which requires a smaller amount of labeling per flow agumentation. Since it is not easy to forecast the sizes of the subnetworks dealt with in the decomposition algorithm, it is hard to determine the overall efficiency of Algorithm A. In any case, Algorithm A is always theoretically more efficient than the efficiency given in Chapter II. At this point we suggest that the empirical efficiency of the proposed algorithm (Algorithm A) be determined by a computer simulation as another research study.

The higher order term $O(mu^3)$ in the shortest path Algorithm C is due to the operations performed on the subnetworks to obtain the conditional distances. For further reduction in those terms, the point of attack must be to find a new algorithm to obtain the conditional shortest distances in the subnetworks in less than $O(mu^3)$ operations.

The only information needed for the construction of the G' network is the conditional shortest distances $D_{X_i,X_{i+1}}$ and $D_{X_{i+1},X_{i+1}}$, i=1, ... , m-2. Thus, any algorithm which can determine these conditional shortest distances in less than $O(mu^3)$ will change the order of operations to a smaller polynomial.

We also suggest that the empirical efficiencies of the shortest path and minimal cost flow algorithms (Algorithms C, D and E) be determined and compared with the empirical efficiencies of the existing decomposition and no-decomposition algorithms.

A large portion of the CPU time required for a no-decomposition algorithm is due to the "triple operations" performed on the nonexisting arcs of a network. This search can be avoided by using a listing of the arcs, but listing itself usually takes more time than the shortest path procedure. Therefore, by decomposing the sparse network into subnetworks we are reducing the number of operations performed on the nonexisting arcs. Any sophisticated programming technique which can be used for a no-decomposition algorithm can also be used on the subnetworks of the proposed decomposition algorithms. The feasibility of such programming techniques must be evaluated before being performed on the decomposition algorithm.

One of the most important issues in network decomposition is the "overhead" involved in the decomposition process. All the comparisons made in this dissertation assumed that the network has already been decomposed. Decomposing the network into M subnetworks will require substantial amount of computer time. Therefore, this aspect of network decomposition should be considered before any attempt is made for using

a decomposition algorithm.

BIBLIOGRAPHY

1. Bazaraa, M. S., and J. J. Jarvis, Linear Programming and Network Flows, John Wiley and Sons, Inc., N.Y., N.Y., 1977.

2. Bazaraa, M. S., and R. W. Langley, "A Dual Shortest Path Algorithm," SIAM Applied Math., Vol. 26, No. 3, 496-501, 1974.

3. Bennington, G. E., "An Efficient Minimal Cost Flow Algorithm," Management Science, Vol. 19, No. 9, 1042-1051, 1973.

4. Berge, C., and A. Ghouila-Houri, Programming, Games and Transportation Networks, John Wiley and Sons, Inc., N.Y., 1966.

5. Brucker, P., "A Decomposition Algorithm for Shortest Pahts in a Network with Many Strongly Connected Components," Zeitschrift for Operations Research, Vol. 18, 181-186, 1974.

6. Cremeans, J. E., R. A. Smith, and G. R. Tyndall, "Optimal Multi-Commodity Network Flows with Resource Allocation," Naval Research Logistics Quarterly, Vol. 17, 269, 1970.

7. Dantzig, G. B., "All Shortest Routes in a Graph," Technical Report 66-3, Operations Research House, Stanford University, California, 1966.

8. _____, Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey, 1963.

9. Dijkstra, E. W., "A Note on Two Problems in Connection with Graphs," Numer. Math., Vol. 1, 395-412, 1969.

10. Dreyfus, Stuart E., "An Appraisal of Some Shortest Path Algorithms," Operations Research, Vol. 17, 395-412, 1969.

11. Edmonds, J., and R. M. Karp, "Theoretical Improvement in Algorithmic Efficiency for Network Flow Problems," JACM, Vol. 19, No. 2, 248-264, April 1972.

12. Erdos, P., and G. Katona, Theory of Graphs, Proceedings of the Colloquium Held at Tihony, Hungary (Academic Press, N.Y., N.Y., 1968).

13. Florian, M., and P. Robert, "A Direct Search Method to Locate Negative Cycles in a Graph," Management Science, Vol. 17, No. 5, 307-310, 1971.

BIBLIOGRAPHY (continued)

14. _____, "Rejoinder: Direct Search Method for Finding Negative Cycles," Management Science, Vol. 19, No. 3, 335-336, 1972.

15. Floyd, R. W., "Algorithm 97, Shortest Path," Comm. of the ACM, Vol. 15, No. 6, 345, 1962.

16. Ford, L. R., and D. R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, N.J., 1962.

17. Frank, H., and I. Frisch, Communication, Transmission and Transportation Networks, Addison-Wesley Co., Reading, Mass., 1971.

18. Glover, F., D. Klingman, and A. Napier, "A Note on Finding All Shortest Paths," Trans. Sci., 3-12, 1973.

19. Glover, F., D. Karney, and D. Klingman, "The Augmented Predecessor Index Method for Locating Stepping-Stone Paths and Assigning Dual Prices in Distribution Problems," Trans. Sci., Vol. 6, No. 2, 171-179, 1972.

20. Gomory, R. E., and T. C. Hu, "Multiterminal Network Flows," J. SIAM, Vol. 9, No. 4, 551-570, 1961.

21. Grigoriadis, M. D., and W. W. White, "A Partitioning Algorithm for the Multicommodity Network Flow Problem," Math Programming, Vol. 3, 155-177, 1972.

22. Gupta, R. P., "On Flows in Pseudosymmetric Networks," J. SIAM Applied Math., Vol. 14, No. 2, 125-225, 1966.

23. Harris, B., Graph Theory and Its Application, Academic Press, N.Y., N.Y., 1970.

24. Hartman, J. K., and L. S. Lasdon, "A Generalized Upper Bounding Algorithm for Multicommodity Network Flow Problems," Networks, Vol. 1, 333-354, 1972.

25. Hu, T. C., "Decomposition Algorithm for Shortest Paths in a Network," J. SIAM, Vol. 15, No. 1, 207-218, 1967.

26. _____, "A Decomposition Algorithm for Shortest Paths in a Network," Operations Research, Vol. 16, No. 1, 91-102, 1968.

27. _____, Integer Programming and Network Flows, Addison-Wesley Co., Reading, Mass., 1969.

28. Jarvis, J. J., and J. B. Tindall, "Minimal Disconnecting Sets in

BIBLIOGRAPHY (Continued)

Directed Multicommodity Networks," Naval Research Logistics Quarterly, Vol. 19, 681-690, 1972.

29. Jarvis, J. J., "On the Equivalence Between the Node-Arc and Arc-Chain Formulations for the Multicommodity Maximal Flow Problems," Naval Research Logistics Quarterly, Vol. 16, 515-529, 1969.

30. Jensen, P. S., "Optimum Network Partioning," Operations Research, Vol. 19, No. 3, 916-932, 1971.

31. Johnson, D. A., "A Note on Dijkstra's Shortest Path Algorithm," JACM, Vol. 20, No. 3, 385-388, 1973.

32. Kant, E., "A Multicommodity Flow Problem," Networks, Vol. 4, 267-280, 1974.

33. Lau, R., R. C. Persiano, and P. P. Variaya, "Decentralized Information and Control: A Network Flow Example," IEEE Trans. on Aut. Control, Vol. AC-17, No. 4, 466-473, 1972.

34. Lin, P. M., and B. J. Leon, "Improving the Efficiency of Labeling Algorithms for Maximum Flows in Networks," JACM Trans., Vol. 19, 1973.

35. Megiddo, N., "Optimum Flows in Networks with Multiple Sources and Sinks," Math. Prog., Vol. 7, 97-107, 1974.

36. Minty, G., "A Comment on the Shortest Route Problem," Operations Research, Vol. 5, 724, 1957.

37. Moore, E. F., "The Shortest Path Through a Maze," Proceedings of an International Symp. on the Theory of Switching, Pt. II, April 2-5, 1957, The Annals of the Computing Laboratories of Harvard University 30, Harvard University Press, Cambridge, Mass.

38. Orden, A., "The Transhipment Problem," Management Science, No. 2, 276-285, 1956.

39. Pape, U., "Implementation and Efficiency of Moore-Algorithms for The Shortest Route Problem," Math. Prog., Vol. 7, 212-222, 1974.

40. Rosentiehl, P., Theory of Graphs, International Symposium, Gordon and Breach, N.Y., N.Y., 1967.

41. Ruefli, T. W., "Decentralized Transhipment Networks," Operations Research, Vol. 19, 1019-1031, 1971.

42. Shea, P. D., On the Detection of Negative Cycles in a Graph, M.S.

BIBLIOGRAPHY (Continued)

Thesis, Georgia Institute of Technology, 1977.

43. Shier, D. R., "A Decomposition Algorithm for Optimality Problems in Tree-Structured Networks," Discrete Math., Vol. 6, 175-189, 1973.

44. Srinivasan, V., and G. L. Thompson, "Accelerated Algorithm for Labeling and Relabeling of Trees with Application to Distribution Problems," JACM, Vol. 19, No. 4, 712-726, 1972.

45. Steenbrink, A. P., Optimization of Transport Networks, John Wiley and Sons, Inc., N.Y., 1974.

46. Swoveland, C., "A Two-Stage Decomposition Algorithm for a Generalized Multicommodity Flow Problem," INFOR, Vol. 11, No. 3, 232-243, 1973.

47. Weintraub, A., "The Shortest and the K-Shortest Routes as Assignment Problems," Networks, Vol. 3, 61-73, 1973.

48. DeWerra, D., "Decomposition of Bipartite Multigraphs into Matchings," Zeitschrift for Operations Research, Vol. 16, 85-90, 1972.

49. Williams, T. A., and G. P. White, "A Note on Yen's Algorithm for Finding the Length of All Shortest Paths in N-Node Nonnegative Distance Networks," JACM, Vol. 20, No. 3, 389-390, 1973.

50. Wollmer, R. D., "Multicommodity Networks with Resource Constraints: The Generalized Multicommodity Flow Problem," Networks, Vol. 1 245-263, 1972.

51. Yen, J. Y., "On Hu's Decomposition Algorithm for Shortest Paths in a Network," Operations Research, Vol. 16, 91-102, 1968.

52. _____, "An Algorithm for Finding Shortest Routes From all Source Nodes to a Given Distination in General Networks," Quarterly Appl. Math., Vol. 27, No. 4, 526-530, 1970.

53. _____, "A Modified 1/2 N^3-Steps Algorithm for Detecting Negative Loops or Finding all Shortest Routes From a Fixed Origin in N-Node General Networks," Paper presented at the TIMS 10th Amer. Meeting, Atlanta, Georgia, Oct. 1-3, 1969.

54. Zadeh, N., "Theoretical Efficiency of the Edmonds-Karp Algorithm for Computing Maximal Flows," JACM, Vol. 19, No. 1, 184-192, 1972.

55. Zionts, S., Linear and Integer Programming, Prentice-Hall, Englewood Cliffs, N.J., 1974.

VITA

Süleyman Tüfekçi was born in Bandırma, Turkey, on January 1, 1948. He is the son of Nihal and Ismail Tüfekçi. He was graduated from Şehit Mehmet Gönenç High School, Bandirma, Turkey, in September 1966.

After four years of undergraduate work at the Middle East Technical University, Ankara, Turkey, he was awarded the Degree of Bachelor of Science in Industrial Engineering in 1971. Subsequently, on September 1973 he was awarded the Degree of Master of Science in the same department.

In October 1973 he came to the United States. He was awarded the Degree of Doctor of Philosophy in Industrial Engineering at Georgia Institute of Technology, in December 1977.

Süleyman Tüfekçi is single and presently residing in Atlanta.